SISTEMAS DIGITALES Y ARQUITECTURA DE COMPUTADORAS

Electricidad;
Electrónica;
Sistemas Numéricos;
Circuitos, Familias y Elementos Lógicos;
Unidad Central de Proceso y Memoria;
Microprogramación y ensamblador;
Dispositivos Externos;
Otros Elementos Lógicos y Electrónicos

2nda edición

EMILIANO LLANO DÍAZ

Exa Ingeniería, S.A. de C.V.

SISTEMAS DIGITALES Y ARQUITECTURA DE COMPUTADORAS
Emiliano Llano Díaz

El autor y Exa Ingeniería® no están afiliados a ningún fabricante.

Compuesto totalmente en computadora por:

Exa Ingeniería SA de CV
Bajío 287-101
Col. Roma México, D.F.
+52 55 5564-10-11
+52 55 5564-02-68
FAX +52 55 5264-61-08

ISBN 970-91050-0-0
SEP 10137/91

Registrado ante la SEP en la propiedad intelectual del autor

Impreso y hecho en México.

1era edición junio 1993.
2da edición junio 2022.

Índice

Índice de figuras

Introducción

1era Edición

Vivimos en la era de la información excesiva. Gracias a los milagros tecnológicos del siglo XX, los ciudadanos del mundo gozamos del acceso instantáneo a más información que la que cualquiera de nosotros podemos captar.

Se puede encontrar información de todo tipo en cantidades que rebasan la imaginación: información de cualquier tema que hayamos pensado (o no) y hasta información de la información.

Es evidente que nuestra era actual requiere de un dispositivo tecnológico dedicado exclusivamente a almacenar, clasificar, comparar, combinar y presentar información a alta velocidad. Tal equipo es la computadora u ordenador.

Ello explica por qué las computadoras aparecen por todos lados que se requiere manejar información; desde en un centro de cómputo gigantesco hasta en una lavadora de ropa o un reloj de pulsera.

Desde el ábaco de los chinos, pasando por los huesos de Napier, la máquina de sumar de Schickard, la de Pascal, la de Leibniz, el telar de Jacquard, la máquina de diferencias de Babbage, las tarjetas de Hollerith, la computadora MARK I hasta las modernas microcomputadoras, aún nos queda un largo camino que recorrer.

Comience por comparar lo que una pequeña abeja puede hacer con las tareas que las más poderosas supercomputadoras que existen no pueden realizar y agregue el hecho de que el cerebro de una abeja tiene aproximadamente 1 millón de neuronas contra 100,000,000,000 que el cerebro humano tiene. Y luego considere la siguiente información:



John Napier
(1550-1617)

Matemático escocés que inventó los logaritmos e introdujo la coma decimal.



Wilhem Schickard
(1592-1635)

Inventor de la primera máquina de calcular que podía multiplicar, restar, sumar y dividir. Murió de la peste y su invento se reconstruyó en una maqueta, a partir de sus bocetos, que actualmente se puede ver en el museo de la ciencia en Múnich, Alemania.



Blaise Pascal
(1552-1662)

Científico y filósofo religioso francés. Estableció los fundamentos de la teoría moderna de las probabilidades, inventó el triángulo matemático, descubrió las propiedades de la cicloide, adelantó el cálculo infinitesimal y formuló la ley que lleva su nombre.



Gottfried Wilheim Leibniz
(1646-1716)

Filósofo y matemático alemán erudito en ciencia, historia y derecho. Desarrolló el cálculo infinitesimal sin conocer los trabajos de Newton al respecto.



Joseph Marie Jacquard
(1752-1834)

Inventor del telar automático que usaba tarjetas perforadas.

- **Eficiencia en el uso de energía**. El cerebro de una abeja disipa alrededor de 10 micro watts (10^{-6}); es mejor en aproximadamente 7 órdenes de magnitud que el circuito más eficiente fabricado hoy en día.
- **Velocidad**. La abeja realiza (calculado en forma tosca y aproximada) unas 10,000,000,000,000 operaciones por segundo (10,000 GFLOPS) mientras que las más potentes computadoras realizan 1,000 veces menos operaciones por segundo.
- **Comportamiento**. Las abejas, como todos saben, liban de las flores almacenando el néctar y polen para regresarlo al panal y hacer miel. Maximizan los beneficios y minimizan sus gastos de energía recordando los mejores sitios y comunicándolo al resto del panal y evitando las flores que ya visitaron. Las abejas pueden ver, volar, caminar y mantener el equilibrio. Pueden navegar grandes distancias y predecir cambios en la distribución de néctar. Reconocen a los intrusos y los atacan; remueven la basura y a las abejas muertas de su colonia y, cuando hay sobrepoblación, una parte emigra a buscar mejores oportunidades en otros sitios alejados.
- **Autonomía y auto dependencia**. Las abejas manejan todas las actividades enumeradas anteriormente sin la necesidad de un ser superior que las supervise mientras que una computadora necesita personal de mantenimiento, supervisión y programación.
- **Tamaño**. El espacio que ocupa el cerebro de una abeja es de unos cuantos milímetros cúbicos. Una maravilla de miniatura. La más pequeña supercomputadora ocupa un área de 2 a 4 metros cúbicos (1995).

Partiendo de esta comparación, parece que tenemos que esperar todavía muchos años para poder hacer que una computadora realice las cosas más sencillas, que todos los días hacemos de forma rutinaria. La naturaleza y sus criaturas son modelos de las formas en que podemos mejorar nuestras arquitecturas actuales.

Desde la computadora manual, pasando por la ENIAC (proceso en secuencia) llegando a las modernas técnicas de computadoras paralelas, interconexión masiva y redes de neuronas tenemos aún mucho camino que recorrer.

Analizamos a lo largo del libro los elementos necesarios para comprender los elementos formativos de una computadora, así como su interconexión, programación y arquitectura. Sentaremos las bases para una comprensión de los procesos internos de las máquinas llevándonos a un mejor aprovechamiento de sus recursos, de las arquitecturas y componentes futuros.

Las computadoras han llegado para quedarse (por lo menos en un futuro mediato) y será cada vez más necesario para los profesionales en el área, entender su funcionamiento; ya no como una caja negra que realiza procesamiento, sino en su forma más íntima.

Los conceptos explicados en la arquitectura de sistemas y programación a nivel circuito son detallados en una microcomputadora, pero sirven por igual, con ligeras modificaciones, para todo tipo de sistema computacional.

Este libro es una recopilación de muchos otros (especificados en la bibliografía de cada capítulo) y de material propio desarrollado a lo largo de 12 años de experiencia profesional con las microcomputadoras, así como 10 de enseñanza en la Universidad Nacional Autónoma de México, en especial en su plantel Acatlán de donde agradezco a los alumnos por sus valiosas contribuciones que hicieron posible este libro, así como por su paciencia ayudando a mi mejoramiento personal y profesional.

Es imprescindible para llegar a conocer el tema a fondo o con la profundidad que nuestro campo así lo demanda, el contacto con muchos otros libros y el mantenerse actualizado, pues el campo de la computación y electrónica está en pleno auge y los cambios se suceden uno tras otro con tal rapidez que el que se pierda de la información, aun por cortos periodos, queda inmediatamente desactualizado. Doy por eso una pequeña bibliografía al final de cada capítulo que, lejos de ser exhaustiva, es la base para una consulta que redunde en más conocimientos de la materia. Es importante también, resolver cada uno de los problemas propuestos y ampliar este trabajo con ejemplos que el profesor diseñe y extraiga de otros libros.

Muchísimas gracias a María Cristina Vera Aristi y a Rubén Romero Ruíz por sus sugerencias y revisión del texto.

México, D.F. a 30 de noviembre de 1992. Emiliano Llano Díaz.



Charles Babbage
(1792-1871)

Matemático británico que trabajó en las primeras tablas de actuaría y planeó una computadora, antecesora de las modernas computadoras, que no finalizó.



Herman Hollerith
(1860-1929)

Inventor americano de la máquina de tabular, predecesora importante de la computadora, usada en el censo de 1890 en EE.UU. Fundó la compañía TBC (Tabulating Machine Company) que más tarde se convirtió en IBM (International Business Machines Corporation).

2da Edición

Han pasado ya casi 30 años desde la 1era edición de este libro. Tantos que ahora vivimos el siglo XXI y los “milagros” tecnológicos siguen siendo espectaculares. Desde su 1era edición ha sido usado por muchas generaciones de alumnos en la UNAM como referencia, libro de texto y de consulta. Su inclusión gratuita en Internet ha dado una amplia difusión mundial a su contenido.

Esta segunda edición debe su nacimiento por sugerencia de varios amigos, alumnos y colegas que insistieron en que, debido a su importancia y difusión, valía la pena una actualización de sus conceptos y contenido.

En esta segunda edición, amplío los temas, corrijo el estilo y su presentación; actualizo y pongo al día la bibliografía que agrupo al final del libro en lugar de en cada capítulo. Aprovecho para cambiar las imágenes y muchos de los ejercicios. Sin embargo, no quise abultar su contenido con ecuaciones o material innecesario, sino mantener lo indispensable en cada unidad. Si el lector requiere ampliar un concepto, existen millones de páginas de libros especializados en cada tema. Tengo la esperanza de haber logrado mi objetivo.

Espero que el lector aproveche las ventajas que la tecnología ofrece para la difusión de los amplios conocimientos que la humanidad ha acumulado y sepa discernir entre lo verdadero y lo ridículamente falso. No porque algo se publique, es verdad.

“El conocimiento que no se comparte se pierde”

Emiliano Llano Díaz.
Junio 2022

1 Corriente Eléctrica

1.1 Los Electrones

La unidad constituyente más pequeña de la materia es el átomo (del latín atŏmus o indivisible — nombrado así por Demócrito de Albera en el siglo V A.C.), y cada uno de ellos contiene un centro relativamente grande o núcleo formado por partículas llamadas protones y neutrones rodeado de pequeños corpúsculos que lo orbitan llamados electrones. Cada uno de estos componentes tiene una propiedad llamada ***carga eléctrica*** que crea una fuerza de atracción y los mantiene unidos. Nótese que el átomo si se puede dividir y no toda la materia del universo está compuesta de átomos y que el modelo del átomo explicado aquí es muy básico (ver el modelo de Schrödinger o de Dirac).

El electrón puede considerarse como el portador más pequeño de carga eléctrica y por convención a esta carga se le considera negativa teniendo una magnitud fija de 1.602 x 10^{-19} ***Coulomb*** o ***Culombios*** (que denotamos por la letra ***C***). Por lo que si denotamos con *e* la carga eléctrica tenemos que:

$$e = 1.602\,x\,10^{-19} \quad C \tag{1.1}$$



Charles Agustín de Coulomb
(1736-1806)

Físico francés que descubre, entre otras, la **Ley de Coulomb** que dice que la fuerza entre dos cargas eléctricas es proporcional al cuadrado de las distancias entre ellas.

La carga eléctrica asociada con el núcleo se debe a los protones (los neutrones no tienen carga) cada uno de ellos con carga *+e* de igual magnitud, pero signo contrario que la de los electrones. En un átomo normal las cargas positivas de los protones están balanceadas por las negativas de los electrones, así que la carga total del átomo es siempre cero. Una materia que no tiene carga eléctrica se le conoce como eléctricamente ***neutra***. Si se le agregan de alguna forma electrones a una substancia neutra, tendrá una carga negativa; si se le remueven electrones se cargará positivamente.

Algunas substancias como el ámbar (cuyo nombre en griego es elektron) y muchos plásticos modernos pueden ser cargados por frotamiento. Por ejemplo, al frotar una regla de plástico contra nuestro cabello transferimos una carga eléctrica que hace que nuestro cabello quede cargado positivamente y la regla negativamente. Si luego acercamos la regla a pequeños pedazos de papel o hacia el cabello de otra persona, podemos ver que estos son

atraídos hacia la regla. Lo mismo sucede al caminar sobre una alfombra de plástico en un día seco o al frotarnos contra asientos de vinilo en un viaje en auto, nos cargamos positivamente y al tocar cualquier parte metálica o a otra persona sufrimos una pequeña descarga eléctrica. Esto es una consecuencia del hecho fundamental de que cargas del mismo signo ya sean positivas o negativas se repelen y cargas de distinto signo se atraen. Podemos resumir lo anterior en la siguiente ley:

(1.2) Cargas distintas se atraen; cargas iguales se repelen

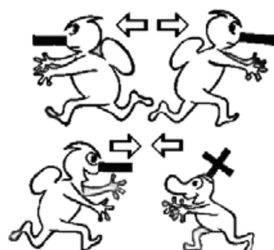


Figura 1.1 Cargas distintas se atraen; cargas iguales se repelen.

Se puede pensar que los electrones se mueven en órbitas circulares fijas alrededor del núcleo (esto es una simplificación; para más detalle consulte la Teoría Cuántica) muy similar a cómo los planetas tienen órbitas con respecto al Sol. Las fuerzas de atracción de dos cuerpos con cargas distintas decrecen al incrementar las distancias. Así, los electrones que se encuentran más alejados del núcleo son los que más fácilmente pueden quitarse del átomo por fricción u otra fuerza externa. Si bien todos los átomos están constituidos por electrones y protones, el número de estos corpúsculos varía cuando los átomos son diferentes. La agitación térmica de los átomos en materiales sólidos, aún a temperatura ambiente, crea una expulsión de electrones de las órbitas externas. Estos electrones comienzan a moverse a través del sólido repeliéndose entre ellos y ocasionalmente son atraídos para ocupar las órbitas disponibles de otros átomos que han quedado con carga positiva por la falta de uno o más electrones. Las substancias con una gran cantidad de electrones libres a temperatura ambiente son conocidos como ***conductores***. En otras substancias, los electrones se hayan fuertemente unidos a su núcleo y se requiere de gran fuerza para poderlos mover. Las materias que están formadas principalmente de átomos de esa clase se les conoce como ***aislantes*** o aisladores. Existen también otro tipo de materiales con propiedades conductoras intermedias entre conductor y aislante y se les conoce como ***semiconductores***. Casi todo dispositivo moderno se

fabrica de los tres tipos de materiales combinados de forma funcional.

Las substancias conductoras más usadas las forman el grupo de los elementos llamados de ***transición*** y en especial los del grupo B1 de la tabla de los elementos químicos de Mendeléyev (cobre, plata y oro) de donde el cobre (Cu) es el más común. Los elementos semiconductores son usualmente los del grupo A III y A IV donde el silicio (***Si***) y germanio (***Ge***) son los más usados y en menor cantidad el selenio (***Se***), galio (***Ga***) y otros (boro, indio, arsénico, antimonio, etc.). Las substancias aislantes son principalmente los plásticos y las baquelitas.



Dmitri Mendeléyev
(1834-1907)

Químico ruso conocido por su descubrimiento del patrón subyacente que llevó a clasificar a los elementos químicos, naturales y creados artificialmente en lo que ahora se conoce como la tabla periódica de los elementos químicos

(1642-1721)

Físico y filósofo inglés famoso por sus experimentos con la luz, la ley de gravitación y las leyes del movimiento.

1.2 El Voltaje

La fuerza de atracción o repulsión entre cuerpos cargados eléctricamente puede medirse en las unidades estándar de fuerza mecánica: los ***Newtons***. Suponga que una fuerza de ***F*** newtons se aplica a una partícula cargada con ***q*** culombios en un punto ***p***. Existe un campo eléctrico ***E*** en el punto ***p*** que se define por la ecuación de fuerza:

$$F = qE \;\; NC \qquad (1.3)$$

Nótese la analogía de la ecuación 1.3 con la ecuación de la fuerza mecánica (segunda Ley de Newton):

$$F = mA \;\; N \qquad (1.4)$$

donde ***m*** denota la masa y ***A*** la aceleración. Así el campo eléctrico ***E***, medido en newtons x culombios, mide la fuerza de aceleración ejercida por las partículas cargadas.



Alessandro Volta
(1745-1827)

Físico italiano inventor de un generador de cargas eléctricas por inducción y la pila que lleva su nombre: pila voltaica.

Para mover un cuerpo cargado eléctricamente a través de una región del espacio que contiene un campo eléctrico, se debe realizar un trabajo (positivo o negativo). El trabajo realizado para mover una carga eléctrica de un culombio del punto ***a*** al punto ***b*** se le conoce como diferencia de potencial o, con más frecuencia, voltaje del punto ***a*** al punto ***b***, y se le denota como $\boldsymbol{v_{ab}}$. Es obvio que

$$v_{ab} = -v_{ba} \tag{1.5}$$

Las unidades del voltaje son los voltios y se definen como el trabajo requerido para llevar una carga eléctrica de 1 culombio a una distancia de 1 metro en un campo eléctrico de 1 newton por culombio. Por lo que 1 volt es igual a 1 newton-metro por culombio o 1 joule por culombio. Si se toma al punto *a* como referencia para la medición del voltaje podemos referirnos a $\boldsymbol{v_{ab}}$ como $\boldsymbol{v_b}$ y como $v_{aa} = v_a = 0$ se dice que el punto ***a*** está a potencial cero, cero volts o ***tierra***.

1.3 La Corriente Eléctrica

Suponga que los bornes de una batería se conectan a los extremos de un buen conductor eléctrico como puede ser un alambre de cobre. Esto crea un campo eléctrico en los conductores que hace que los electrones libres cargados negativamente se muevan al extremo positivo de la batería. El flujo de electrones o de otras partículas cargadas eléctricamente de esta forma se le conoce como ***corriente eléctrica***.

Si esta conexión se realiza sin poner de por medio una carga que absorba parcialmente la gran cantidad de electrones en movimiento, se crea lo que se conoce como ***corto circuito*** y la batería se descarga totalmente o se quema el cable usado para la conexión (este principio se usa en los fusibles para proteger las instalaciones). Si los extremos del alambre no se tocan, los electrones no pueden pasar y a esto se le conoce como ***circuito abierto*** (principio aplicado en los interruptores eléctricos).

La unidad usada para medir la corriente eléctrica (flujo de electrones) es el ***Amper*** o Amperio. Un amperio corresponde a 1 culombio de carga pasando por un punto de medición por segundo. Usando la ecuación (*1.1*) esto es equivalente a un flujo de 1.602 x 10^{19} electrones por segundo, aproximadamente cuatro veces la corriente necesaria para prender una lámpara de linterna. La instalación casera típica se protege con fusibles de 30 amperes y es raro que se fundan a menos de que se sobrecarguen las líneas o se haga un corto circuito. La corriente utilizada en circuitos electrónicos es mucho menor y se mide típicamente en mili (milésima parte) o microamperios (millonésima parte de amperio)[1].



André Marie Ampère
(1775-1836)

Físico, filósofo naturista y matemático francés que trabajó en electrodinámica estableciendo la ley que lleva su nombre. Estudió las relaciones entre electricidad y magnetismo.

1.4 Circuitos Eléctricos

La mayoría de los sistemas descritos en este libro son circuitos eléctricos y electrónicos. Sus componentes básicos son:

- Una Fuente de Poder.
- La Interconexión entre los componentes usando conductores eléctricos.
- Una Carga.

La figura *1.2* muestra un esquema estándar y un circuito eléctrico común formado por una batería y una lámpara.

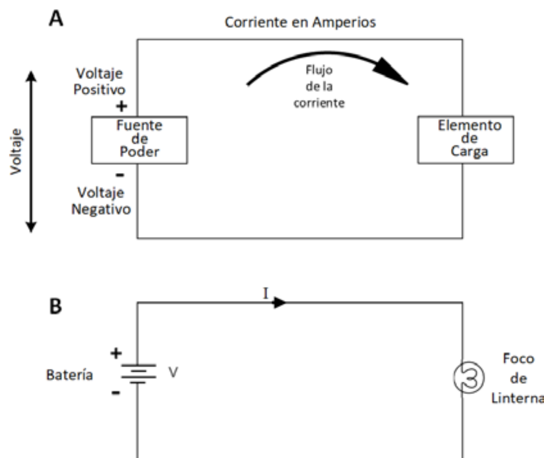


Figura 1.2 Esquema estándar y circuito eléctrico.

[1] Peligro de muerte por electrocución = 100mA; Umbral de sensación del cuerpo humano = 3,000 µA

1.4.1 Fuentes de Poder



James Prescott Joules
(1818-1889)

Físico inglés que fue el primero en determinar el equivalente mecánico del calor.

La potencia de la fuente de poder mantiene una diferencia de potencial entre sus terminales y provee de la energía necesaria para mover las cargas eléctricas por el circuito. Si ***q*** culombios pasan por el circuito por unidad de tiempo, se deben restituir ***qV*** Joules que usa la carga constantemente.

Una pila logra esto usando la energía química acumulada en sus celdas. Otra forma de obtener corriente eléctrica es a partir de la luz solar (celdas fotovoltaicas). En nuestras casas la compañía de luz nos entrega la potencia que gastamos en nuestros distintos aparatos domésticos (cargas eléctricas) y que produce de forma electromagnética usando la fuerza de una caída de agua, quemando combustible o carbón, con la fuerza eólica o de las mareas, la energía nuclear u otra. En un automóvil, por ejemplo, usamos la batería que combina plomo y ácido; a su vez recargamos dicha batería quemando combustible y moviendo un generador o alternador[2] (que convierte energía mecánica en eléctrica) con el mismo motor que impulsa al automóvil.



Figura 1.3 Pila seca alcalina.

[2] Un ***alternador*** produce electricidad cuando un campo magnético gira dentro de un estator (bobinas de alambre). El alternador produce corriente alterna que debe, entonces, convertirse a corriente directa para poder utilizarse en el coche. En cambio, en un ***generador*** la armadura o devanados de alambre giran dentro de un campo magnético fijo generando electricidad. Según se cablee el generador, éste producirá ya sea corriente alterna o corriente directa. Más adelante en este capítulo hablaremos de los distintos tipos de corrientes.

Supóngase que una corriente constante de *I* amperes fluye por el circuito de la figura *1.2*. De la definición de corriente y voltaje concluimos que la energía consumida por la carga o generada por la fuente de poder en un segundo es de ***VI*** Joules. Así, la *potencia* ***P*** (medida en Watts o vatios) consumida está definida por la ***Ley de Watt***:



James Watt
(1736-1819)

Inventor escocés de un nuevo tipo de máquina de vapor patentada en 1769.

$$P = VI \quad (W) \tag{1.6}$$

Una analogía práctica se puede realizar entre el circuito de la figura *1.2* y el sistema hidráulico de la figura *1.4*. El flujo de electrones por los alambres[3] de cobre del primer caso corresponde al flujo de agua en el segundo. La diferencia de presión entre dos puntos del sistema hidráulico corresponde a la diferencia de potencial o voltaje (***v***) del circuito eléctrico y la cantidad de agua sería el equivalente a la corriente eléctrica (***i***). En la figura *1.4* la energía es restituida por la bomba (pila, batería) que impulsa de nuevo el agua por la tubería (conexión eléctrica) hacia la rueda de molino (carga) que la remueve parcialmente (consume).

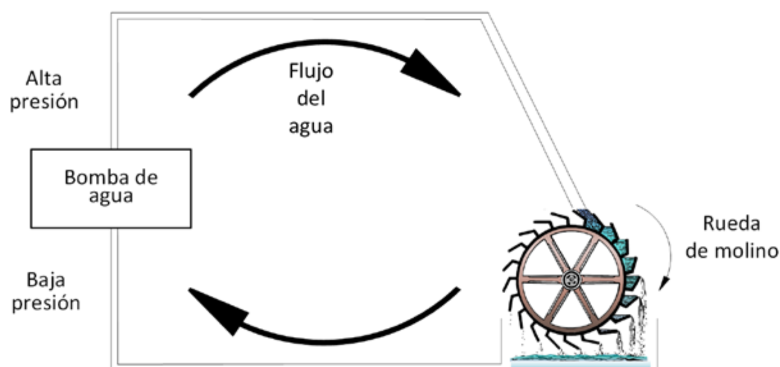


Figura 1.4 Símil hidráulico de un circuito eléctrico.

Investigue

Averigüe que es un dínamo y cómo funciona. Dé al menos un ejemplo de su aplicación práctica en la vida diaria.

Experimente

Use "Alimentos Eléctricos" para crear una batería. Puede inspirarse usando ya sea limones, patatas (papas), monedas de cobre y vinagre, etc.

[3] El alambre es un conductor eléctrico único, mientras que un cable es un grupo de alambres trenzados. Generalmente son de cobre y están envueltos en un revestimiento aislante.

1.4.2 Corriente Directa y Corriente Alterna

Existe una diferencia más entre la energía que nos proporciona una pila, batería (o acumulador)[4] y las que nos puede proporcionar un generador (la compañía de luz o el automóvil). La corriente generada por medio del uso de energía química es constante en el tiempo y disminuye poco a poco hasta agotarse o hasta que sea regenerada para poder producir más corriente (ver la figura *1.4*). A los circuitos que requieren de este tipo de alimentación se les conoce como de ***corriente directa*** o ***corriente continua*** (***CC***, ***CD*** o ***DC*** en inglés).

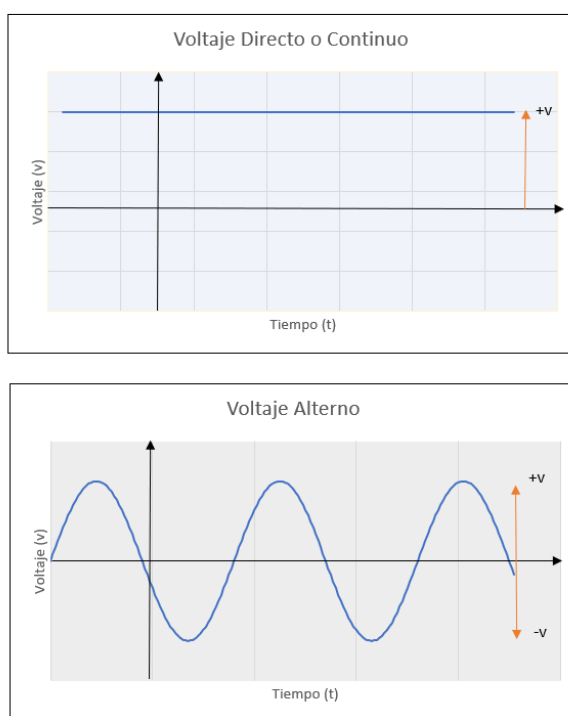


Figura 1.5 Voltaje directo y voltaje alterno.

Al utilizar un generador para crear movimiento de electrones se utiliza un principio donde al cortar un campo magnético con un conductor, el campo genera una corriente de electrones en el conductor. Los imanes y campos magnéticos tienen dos polos por lo que al pasar el conductor por el polo contrario se genera una corriente en sentido inverso. Este cambio de sentido constante

[4] Una pila (alcalina, zinc-carbono, litio) es un dispositivo que convierte energía química en energía eléctrica. Su voltaje va de 1.5 a 9 voltios. No puede recargarse. Una batería (acumulador; níquel-cadmio, litio, plomo) es un dispositivo que convierte energía química en energía eléctrica, pierde energía con el paso del tiempo, pero puede recargarse. Tiene uso doméstico e industrial. Tanto la pila como las baterías entregan corriente directa.

genera una corriente que alterna de positiva a negativa una y otra vez (ver la figura *1.5*). A estos circuitos y a las fuentes de poder que entregan este tipo de voltaje o corriente se les conoce como de ***corriente alterna*** (***CA*** o ***AC*** en inglés). Es interesante hacer notar que la gran mayoría de los generadores son reversibles, esto es, si se hace girar el rotor generan energía eléctrica y si se aplica energía eléctrica al estátor el rotor se mueve (motores eléctricos, vea la figura *1.6*).

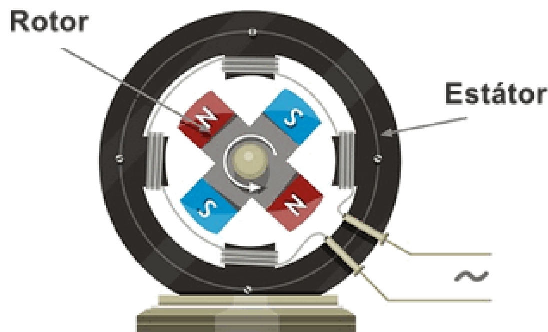


Figura 1.6 Generador o motor de corriente alterna.

Existen otras fuentes de generación de corriente alterna (***CA***) como son los micrófonos, osciladores, fonocaptores, etc. Hay muchas clases de corrientes alternas con distintos tipos de ondas, pero la que más interés tiene es la sinusoidal.

Si examinamos cuidadosamente una onda sinusoidal (la más común de las ondas de ***CA***) veremos que la corriente comienza en cero, que asciende, gradualmente, hasta alcanzar su valor máximo, y entonces retorna a cero para completar la primera mitad del ciclo. Luego, la corriente invierte su sentido de flujo, avanza gradualmente hasta llegar a la cúspide, en la dirección opuesta, y retorna nuevamente a cero para finalizar el ciclo completo. En una onda sinusoidal pura, el valor máximo en ambos sentidos es igual.

Cuando se menciona el valor de una onda sinusoidal, generalmente nos referimos a su valor ***efectivo***, el cual es considerablemente menor que su valor máximo. Para efectuar la conversión de un valor a otro, se usan las siguientes fórmulas aproximadas:

$$V_{efectivo} = 0.7 \; x \; V_{máximo} \tag{1.7}$$
$$V_{máximo} = 1.4 \; x \; V_{efectivo} \tag{1.8}$$

Por consiguiente, cuando decimos que la corriente en nuestra casa tiene un voltaje efectivo de 120 voltios, el pico real es 1.4 x 120v, o sea aproximadamente 168 voltios.

Es posible convertir con relativa facilidad de un tipo a otro de corriente (mucho más fácil de alterna a directa) y casi todos los circuitos electrónicos que analizaremos en este libro (los cuales contienen semiconductores) trabajan exclusivamente con corriente directa.

1.4.2.1 Frecuencia

La repetición cíclica de una señal un número determinado de veces durante un segundo se le llama ***frecuencia***. Dicha frecuencia se representa con la letra (***f***) y su unidad de medida es el ciclo por segundo, Hercio o Hertz (***Hz***). Sus múltiplos más generalmente empleados son los siguientes:

- Kilohertzio (kHz) = mil Hertz
- Megahertzio (MHz) = un millón de Hertz
- Gigahertzio (GHz) = mil millones de Hertz

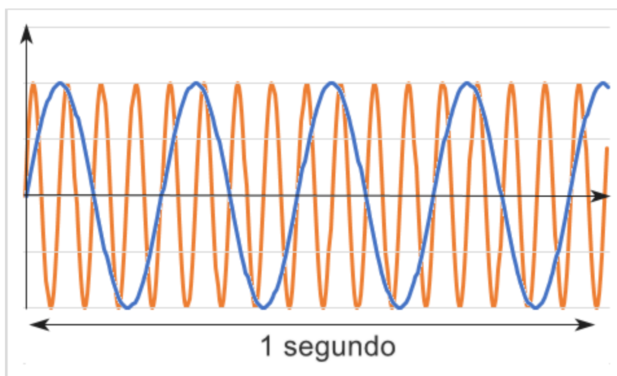


Figura 1.7 Frecuencia de una onda.

La corriente alterna se distribuye a 60 Hertz en América y gran parte de Japón mientras que en el resto del mundo se distribuye a 50 hercios.

1.4.2.2 Periodo

El tiempo necesario para que un ciclo se repita se le conoce con el nombre de ***periodo***. Se mide en segundos y se representa con la letra ***T***.

La frecuencia y el periodo de una señal son valores inversos.

(1.9) $$f = \frac{1}{T} \ Hz$$

1.4.2.3 Amplitud

El máximo valor alcanzado (valor pico en voltios) de la corriente alterna se define como la amplitud de esta. Recordemos que es una onda que varía continuamente en el tiempo. Por ello existen distintos valores a considerar: valor pico, valor eficaz, valor medio y valor máximo.

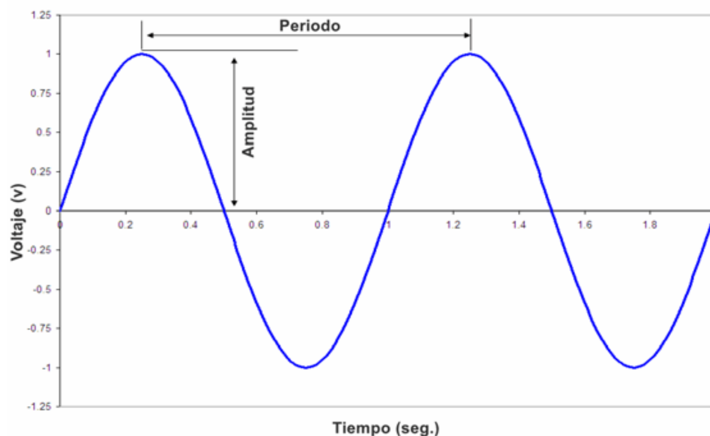


Figura 1.8 Periodo y amplitud de una onda.

Aunque existen distintos tipos de ondas de corriente alterna, el típico usado en la distribución casera (después de su paso por un transformador, descrito en el siguiente capítulo) es una onda senoidal de 120 voltios efectivos[5].

1.4.2.4 Fase

Las ondas senoidales tienen un valor máximo positivo en un tiempo π/2 y un valor máximo negativo en 3π/2. Los valores cero ocurren en la línea base en 0, π y 2π.

Sin embargo, no todas las ondas senoidales atraviesan el eje del tiempo en su punto cero al mismo tiempo. Puede ser que se encuentren desplazadas a la derecha o a la izquierda al compararse con otra onda senoidal.

---

[5] En América y Japón el voltaje que llega a las casas es de 120v efectivo mientras que en Europa y gran parte del mundo se distribuye a 220v efectivos. Generalmente se transmite a 66 kV o más de forma aérea usando cables de aluminio reforzado con acero sin aislante y la mayoría de la corriente transita por el exterior del cable debido al efecto de piel.

Si se encuentra que al comparar dos ondas senoidales este desplazamiento existe, se habla entonces de un *desfase* o diferencia de fase (simbolizado por la letra Φ (letra Griega Phi), expresado en grados o radianes con respecto a un punto de referencia a lo largo del eje del tiempo.

Dicho en otras palabras, el desfase es una diferencia lateral entre dos o más ondas a lo largo de un eje común y puede ir de 0 a su máximo periodo ***T*** ($0 \leq \Phi \leq 2\pi$ (radianes) o $0 \leq \Phi \leq 360^o$).

1.5 Ley de Ohm

Examinaremos un circuito de corriente continua o directa (***CC***), en el que los voltajes y corrientes asociadas con cada elemento del circuito y la fuente son de dirección fija. En muchos casos los parámetros eléctricos son también fijos en magnitud, esto es, no varían en el tiempo. Una fuente de poder de corriente directa tal como una batería puede ser vista ya sea como una fuente de voltaje o de corriente y se representa con el símbolo de la figura *1.6*.

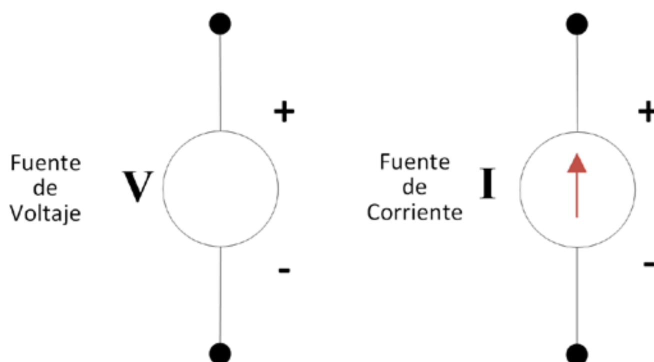


Figura 1.9 Voltaje directo y voltaje alterno.

Las dos terminales de la fuente se marcan como positiva (+) o negativa (-), donde, por convenio, una corriente positiva compuesta por portadores de corriente cargados positivamente emerge de la terminal positiva, pasa por el circuito y regresa por la terminal negativa. Nótese que la corriente *+I* que fluye por los conductores en una dirección es equivalente a la corriente *-I* que fluye por el mismo conductor, pero en sentido contrario.

Imagine que una fuente de voltaje ideal con un voltaje *V* es conectada a una carga de dos terminales ***D*** como se muestra en la figura *1.2*. Hay una relación exacta entre el voltaje, la corriente y la resistencia al paso de corriente que presenta el circuito. Esta relación se le denomina ***Ley de Ohm*** en honor a su descubridor. Si la corriente resultante ***I*** que fluye a través de ***D*** es directamente proporcional al voltaje podemos escribir:

$$v = RI \tag{1.10}$$

A la constante de proporcionalidad que relaciona la corriente con el voltaje se le llama **resistencia eléctrica** (o simplemente resistencia) y se mide en ***Ohmios*** (cuyo símbolo es la letra griega Omega Ω), donde 1 ohmio es la resistencia de un dispositivo que permite el paso de 1 amperio cuando se le aplica un voltaje de 1 voltio. La ley de Ohm se puede aplicar a la gran mayoría de los materiales que estén a una temperatura constante (la resistencia de un material varía con la temperatura). Los materiales conductores tienen una baja resistencia (fracciones de Ohm) mientras que los aislantes tienen gran resistencia (mega ohm). A los elementos diseñados para oponerse al paso de la corriente eléctrica se les llama resistencias y hablaremos de ellas en el siguiente capítulo.

Si se aplica a un resistor (resistencia) una tensión alterna, la corriente, a su paso por la resistencia, variará del mismo modo que el voltaje aplicado. Una onda sinusoidal de voltaje producirá una onda sinusoidal de corriente. La relación que hay entre la corriente, el voltaje y la resistencia, en un circuito de ***CA***, es la misma que en un circuito de ***CC***, y se aplican las mismas fórmulas derivadas de la ley de Ohm.

Al igual que en la ***CC***, la resistencia de un circuito limita la cantidad de corriente que puede fluir como resultado de la aplicación de cierto voltaje. Sin embargo, en un circuito de ***CA*** hay otros dos elementos con los cuales se puede limitar la cantidad de corriente: Los capacitores (condensadores) y los inductores (bobinas) de los que hablaremos en el siguiente capítulo.



George Simón Ohm
(1787-1854)

Físico y matemático alemán. Realiza sus investigaciones con la pila inventada por Alessandro Volta y descubre la relación que lleva su nombre.

Ejercicios

1.1 Un foco (bombilla) de linterna tiene una resistencia de 10 Ohmios (10Ω) y el fabricante indica que se requieren 0.15 amperios (0.5A) para hacerlo funcionar a su máxima intensidad. ¿Qué voltaje debe aplicarse al foco y cuál es la potencia que disipa?

Respuesta
1.5 Voltios @ 225 mili watts (1.5v @ 225mW).

1.2 Un cautín se conecta a un enchufe de corriente de 120 Voltios y con un amperímetro se mide el flujo de corriente que resulta ser 2 amperios. ¿Qué resistencia presenta la punta del cautín al paso de la corriente y que potencia disipa?

Respuesta
55 ohmios @ 240 vatios (55Ω @ 240W).



Gustav Robert Kirchhoff (1824-1887)

Físico alemán que trabajó con Bunsen en el análisis espectral; descubrió el Cesio y el Rubidio. Explicó la presencia de las rayas de Fraunhofer en el espectro solar.



Antoine Laurent Lavoisier (1743-1794)

Químico y físico francés fundador de la química moderna. Determinó la naturaleza de la combustión y el papel del oxígeno en la respiración. Fue guillotinado durante el régimen del Terror por pertenecer a la nobleza.

1.5.1 Leyes de Kirchhoff

El análisis de circuitos eléctricos usualmente consta de una serie de cálculos de voltajes y corrientes en varios puntos del circuito. Este análisis usualmente se basa en dos principios fundamentales conocidos como las leyes de Kirchhoff. La primera de ellas, la ***ley de las Corrientes***[6], establece:

(1.11) *La suma algebraica de las corrientes que entran a un nodo es cero*

Aquí "suma algebraica" significa la dirección o el signo (positivo o negativo) de cada corriente que deba considerarse. La ley es una consecuencia inmediata del hecho que el mismo número de portadores que entran a un nodo deben salir de él; en otras palabras, las cargas eléctricas se conservan (Ley de la conservación de la energía de Lavoisier). Sugiere también que la corriente eléctrica se comporta de forma similar a un líquido.

(1.12) $$\sum_{k=1}^{n} \check{I}_k = 0$$

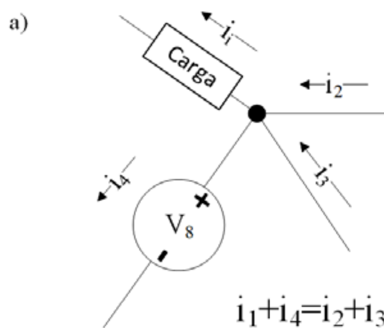
El segundo principio de análisis llamado ***Ley de Voltajes***[7] establece:

[6] Ley de los Nodos o Primera Ley de Kirchhoff.
[7] Ley de las Tensiones o Segunda Ley de Kirchhoff.

(1.13) *La suma algebraica de todas las diferencias de potencial en un circuito cerrado es cero*

Esta ley expresa el hecho de que la energía producida por la fuente de poder es igual a la consumida por los elementos; en otras palabras, la energía eléctrica se conserva.

(1.14) $\sum_{k=1}^{n} \check{V}_k = 0$



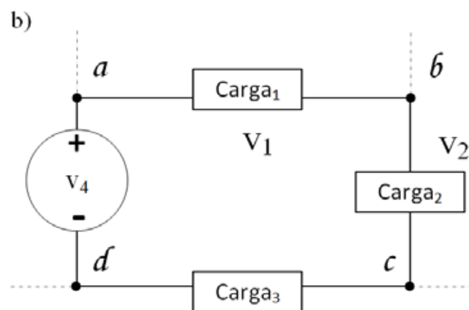
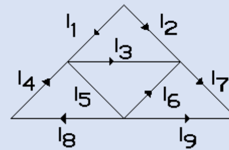


Figura 1.10 Leyes de Kirchhoff.

Como resultado directo de estas leyes, las fuentes de voltaje se pueden conectar en serie (una tras otra) o en paralelo si se requiere aumentar el voltaje o la corriente. Fuentes de voltaje conectadas en serie no aumentan la corriente (1era ley de Kirchoff) pero el voltaje total es la suma de cada uno de los individuales (2da ley de Kirchoff). En cambio, si se conectan en paralelo, el voltaje es el mismo (2da ley de Kirchoff) pero la corriente total disponible es la suma de todas las que puedan proporcionar las fuentes de forma independiente (1era ley de Kirchoff).

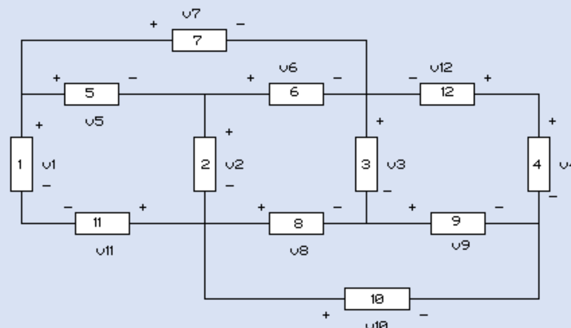
Ejercicios

1.3 Encuentre todos los voltajes del siguiente circuito eléctrico sabiendo que i_1=2, i_3=1, i_7=2 (todas las corrientes en amperios).



b) Encuentre todas las corrientes del mismo circuito si se sabe que v_1=v_3=v_6=v_9=1 (todos los voltajes en voltios).

1.4 Encuentre todas las corrientes y voltajes del siguiente circuito eléctrico sabiendo que v_1=10, v_2=5, v_4=-3, v_6=2, v_7=-3, v_{12}=8, i_1=2, i_7=-5, i_4=5, i_{10}=-3, i_3=1 (todas las corrientes en amperios y voltajes en voltios).





León Charles Thévenin (1857-1926)

Telegrafista francés que extiende la Ley de Ohm para el análisis y reducción de circuitos eléctricos complejos.



Edward Lawry Norton (1898-1983)

Ingeniero Eléctrico que trabajó para los laboratorios Bell. Posee numerosas patentes y es conocido principalmente por su teorema, el dual del de Thévenin.

1.5.2 Teorema de Thévenin y Norton

Muchos circuitos sólo tienen 2 terminales externas. Un circuito de dos terminales puede ser imaginado como una caja negra donde un voltaje ***V*** y una corriente ***I*** aparecen en las terminales externas. Se puede demostrar que un circuito de este tipo tiene dos equivalentes llamados de Thévenin y de Norton. La conexión en serie de una resistencia (invariable en el tiempo y lineal) con una fuente de voltaje (Thévenin) y la conexión en paralelo de una fuente de corriente con una resistencia (Norton) son equivalentes porque presentan las mismas características (vea la figura 1.7). En algunos casos es más conveniente representar un circuito con una fuente de voltaje que con una de corriente. En otras situaciones encontramos más conveniente el uso de fuentes de corriente. Los circuitos equivalentes de Thévenin y Norton nos dan esta flexibilidad.

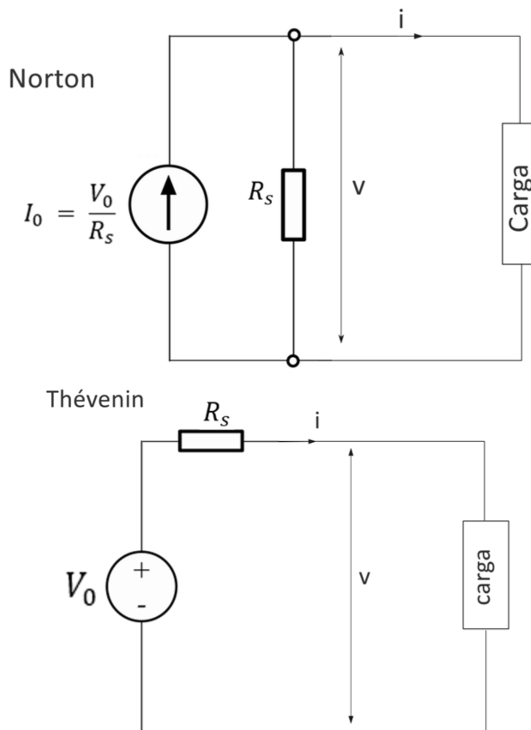


Figura 1.11 Teoremas de Thévenin y Norton.

1.6 Medición

La humanidad se pasa el tiempo tratando de nombrar y medir las cosas. Es común usar instrumentos para contar, medir y pesar cantidades físicas. Pero con cantidades eléctricas que deben medir el paso o flujo de electrones que hasta hace poco desconocíamos o no podíamos ni siquiera ver, esto era muy difícil. No podemos recalcar la importancia de los instrumentos eléctricos de medición ya que mediante el uso de estos auxiliares que indican magnitudes eléctricas tales como la corriente, carga, potencial y energía, o las características eléctricas de los circuitos (que estudiaremos en los próximos capítulos), como la resistencia, la capacidad o inductancia, se pueden medir con precisión dichas unidades. Además de ello, estos instrumentos nos permiten localizar fallas en caso de una operación defectuosa en aparatos eléctricos o electrónicos en los cuales, como es bien sabido, no es posible apreciar su funcionamiento en una forma visual, como en el caso de un aparato mecánico o hidráulico.

Aun en el caso de los equipos de medición más sofisticados ninguno de ellos llega a ser preciso del todo. Para que un instrumento

cumpla su cometido de dar una lectura fiable debe cumplir con las siguientes características[8]:

- La *precisión* es una medida de la veracidad de la lectura. Esto es, el error máximo que ocurre al obtener una lectura.
- La *resolución* es la medida del menor cambio que el instrumento puede registrar o medir. Está relacionada con la sensibilidad del instrumento.
- La *exactitud* es el grado de reproducibilidad.
- La *linealidad* es una medición de la desviación entre la salida del transductor y el desplazamiento real que se mide. Indica si el sistema de medición tiene la misma exactitud para todos los valores de referencia.
- La *sensibilidad* es una medida del cambio en la salida del instrumento que se presenta si la cantidad medida varía en una proporción determinada. Los instrumentos de calidad tienen una sensibilidad máxima de cerca de 1 μV (***CA*** o ***CD***).
- La *tolerancia* se relaciona con la exactitud y precisa el error máximo previsto en algún valor.
- El *rango* indica el intervalo de valores de determinada variable que determinado instrumento es capaz de medir. En los instrumentos analógicos el usuario debe decidir cuidadosamente el rango a medir so pena de quemar o dañar irreversiblemente el aparato de medición. En cambio, en los instrumentos de medida digitales se puede usar un auto rango y luego realizar ajustes.
- La *fiabilidad* capacidad de obtener el mismo resultado en distintas pruebas.
- La *gama* se define como la diferencia entre el máximo y menor valor medible que puede ofrecer el instrumento.

[8] La metrología es la rama de la ciencia que estudia la medición de las magnitudes garantizando su normalización mediante la trazabilidad.

1.6.1 Corriente

Los ***galvanómetros***[9] son los instrumentos principales en la detección y medición de la corriente. Se basan en las interacciones entre una corriente eléctrica y un imán. El mecanismo del galvanómetro está diseñado de forma que un imán permanente o un electroimán (descritos en el siguiente capítulo) produce un campo magnético, lo que genera una fuerza cuando hay un flujo de corriente en una bobina cercana al imán. El elemento móvil puede ser el imán o la bobina. La fuerza inclina el elemento móvil en un grado proporcional a la intensidad de la corriente. Este elemento móvil puede contar con un indicador o algún otro dispositivo que permita leer en un dial el grado de inclinación y, por ende, el amperaje que transita por un circuito.



Jacques-Arsène d'Arsonval (1651-1940)

Biofísico e inventor francés, que ideó el galvanómetro de bobina móvil y el amperímetro termopar. Junto a Nikola Tesla, estudió los efectos de la electricidad en los organismos biológicos.

El galvanómetro se integra, junto con todos los circuitos, diales, botones selectores e indicadores requeridos en un aparato único llamado *amperímetro* que puede ser tanto de tipo analógico como digital. Hoy en día es difícil disponer de modelos analógicos, pero aún existen. Los modelos digitales ofrecen grandes ventajas como son: auto rango, protección contra sobre corriente, mayor precisión, bloqueo de cuenta y muchas otras. Los modelos modernos no sólo leen la corriente, sino que integran (dependiendo el modelo y el precio) el amperaje, voltaje, potencia y muchas más funciones de utilidad y se les conoce con el nombre de *multímetros*.

Así como, para medir el caudal de agua en un circuito hidráulico se intercala el contador en la tubería, para medir las cargas que se desplazan por un circuito en una unidad de tiempo (amperes) el amperímetro se intercala en el conductor donde nos interesa saber la corriente. A esta forma de conexión se le denomina en *serie*.

Al conectarse el amperímetro en serie intercalándolo en el conductor, éste debe presentar la menor carga[10] (llamada resistencia; que analizaremos en el próximo capítulo) posible al movimiento de electrones para no falsear la lectura.

[9] Este término es de uso común desde 1836 y se deriva del apellido del investigador italiano Luigi Galvani, que descubrió que la corriente eléctrica podía hacer mover los músculos de la pata de una rana.

[10] Idealmente cero.

Cabe notar que todo lo descrito anteriormente sólo se aplica a la medición de corriente directa. Si se trata de medir corriente alterna se debe usar otro tipo de medidor llamado de gancho o pinza que puede leer corrientes de 20 a 1,000 o más amperes y funciona tanto para ***CD*** como ***CA***. Estos instrumentos se usan envolviendo, con una de las mordazas del aparato, uno de los cables (sólo un polo) que transportan la energía hacia la carga.



Medir la corriente con uno de estos dispositivos puede ser peligroso y hasta mortal. Si no está seguro de cómo proceder, lea cuidadosamente las instrucciones del aparato o pida consejo de una persona experimentada antes de intentar hacerlo por su cuenta.

Use la regla de la "Mano en el bolsillo": Use un clip (caimán) en el cable o terminal negra del instrumento para conectarlo a la tierra del circuito (-) y, mientras coloca una mano en su bolsillo, use la otra para guiar el cable de prueba rojo (positivo) a los puntos de prueba de interés. Es una metáfora, pero puede salvarle la vida.



Figura 1.12 Contador de agua, amperímetro y galvanómetro. Multímetro digital, analógico y amperímetro de pinza o gancho.

1.6.1 Voltaje

Para medir la tensión eléctrica se precisa de un aparato de medida que sea capaz de leer el desnivel eléctrico o diferencia de cargas de un punto a otro. A este aparato se le denomina *Voltímetro*. El

voltímetro se conecta siempre entre los dos puntos entre los cuales se quiere realizar la medición teniendo la precaución de hacerlo en ***paralelo*** (derivación) como se muestra en la figura *1.13*.

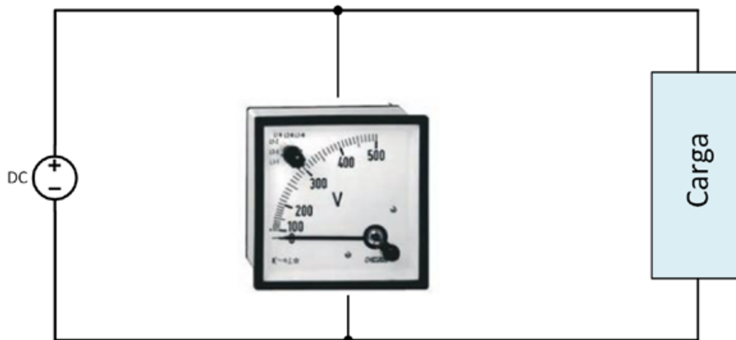


Figura 1.13 Medir el voltaje de un ramal de un circuito eléctrico.

Para que un aparato pueda medir el voltaje de un circuito, no debe producir una carga[11] apreciable en el mismo, o de lo contrario modifica el propio valor de lo que mide.

Las mismas reflexiones de los tipos de instrumentos disponibles que las realizadas para el amperímetro se aplican al voltímetro. Refiérase a la sección correspondiente para una descripción detallada.

1.6.2 Potencia

El aparato usado para medir la potencia eléctrica es el *vatímetro*. En realidad, el vatímetro mide por separado la tensión y la intensidad de la corriente para después realizar la operación $P = VI$. En general no se usa más que para para medir la potencia de equipos conectados a líneas da ***CA***.

Puesto que el aparato debe medir simultáneamente el voltaje como la corriente, deberá conectarse tanto en serie como en paralelo para que el instrumento se encargue de todo tipo de cálculo requerido y lo presente en su pantalla (también existen versiones analógicas de estos instrumentos, pero están completamente en desuso). En la figura *1.14* se puede ver la forma de conectar un vatímetro digital.

[11] Debe tener una elevada resistencia interna para extraer muy poca corriente del medio a medir. Este concepto se expande en el siguiente capítulo.

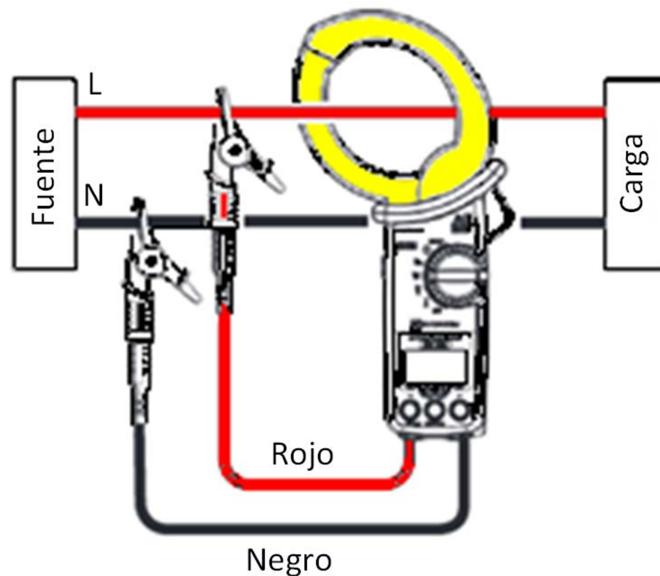


Figura 1.14 Medir la potencia de un circuito eléctrico con un vatímetro digital.

1.6.3 Osciloscopios

Un osciloscopio presenta los valores de las señales eléctricas en forma de coordenadas en una pantalla, en la que normalmente el eje x (horizontal) representa el tiempo y el eje y (vertical) representa la tensión. La imagen que se obtiene así se llama *oscilograma*. Los osciloscopios suelen incluir otra entrada, llamada "eje Thrasher" o "Cilindro de Wehnelt" que controla la luminosidad del haz, permitiendo resaltar o apagar algunos segmentos de la traza. Existen modelos tanto analógicos (en desuso) como digitales. Un osciloscopio puede contar con una entrada (canal) o varias (5 o más). La figura *1.15* muestra un esquema de un osciloscopio digital de un canal.

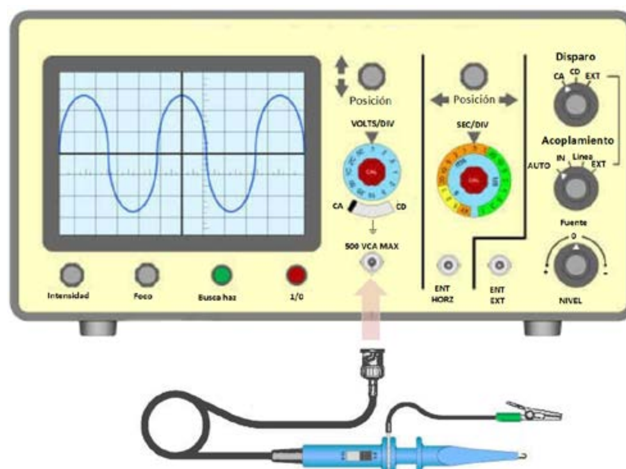


Figura 1.15 Esquema de un osciloscopio digital de un canal.

1.7 Resumen

Este capítulo resulta básico para entender mejor todos los conceptos que se introducen en los siguientes capítulos. Aunque muy bien se puede prescindir de él, la comprensión no sería tan completa. Se introduce el concepto de cargas eléctricas, voltaje, corriente y algunas de las leyes fundamentales de los circuitos eléctricos.

1.8 Puntos Importantes del Capítulo

- Las Cargas Eléctricas crean una fuerza de atracción que mantiene unidos a los componentes del átomo.
- Las Cargas Eléctricas hacen posible que haya electricidad.
- El Voltaje es el trabajo que una carga eléctrica realiza.
- La Corriente es el símil del gasto hidráulico (cantidad de agua por unidad de tiempo) así como el Voltaje de la intensidad de gasto hidráulico (fuerza o presión del agua).
- Los Circuitos Eléctricos se basan en una fuente de voltaje y varios componentes
- Existen dos tipos importantes de corrientes o voltajes: la Directa y la Alterna (***CA*** y ***CD***).
- La Ley de Ohm relaciona el voltaje, la corriente y la resistencia del circuito.
- La Potencia relaciona el voltaje y la corriente y nos da un valor de su consumo o gasto.
- Las Leyes de Kirchoff establecen la conservación de la energía para circuitos eléctricos.
- Los Teoremas de Thévenin y Norton dan equivalencias a las fuentes de voltaje y de corriente.
- Existen varios tipos de instrumentos para medir las corrientes, voltajes y potencias.

1.9 Problemas

1.1 Una batería de auto tiene un voltaje en circuito abierto de 12V. Para arrancar el auto es necesario entregar una corriente de 80A al motor de arranque que puede ser visto como una resistencia de 0.1W. En un día de invierno la resistencia interna de la batería sube de 0.02W a 0.2W. ¿Arrancará el automóvil?

1.2 Investigue que es una ***FEM***.

1.3 Usando el diagrama de la figura *1.2* y suponiendo que el voltaje de la pila es de 1.5 Voltios y la resistencia del foco de 1Ω. ¿Cuál es la corriente, *I*, que circula en el circuito?

1.4 Realice el mismo ejercicio descrito en el problema anterior con los siguientes datos: El voltaje de la pila es ahora de 2 voltios y el consumo del foco del foco sube a 3 vatios (3W). ¿Cuál es la corriente, *I*, que circula en el circuito?

1.5 Explique por qué el voltaje de una pila cae cuando se le aplica una carga a diferencia de la medida del voltaje sin carga.

1.6 Explique cómo funciona un motor eléctrico sencillo. Investigue los distintos tipos que existen y realice un sencillo diagrama de por lo menos 2 de ellos.

2 Elementos Eléctricos y Electrónicos

En un circuito electrónico sencillo tal como el de la figura *1.2* existe siempre una fuente de poder que actúa como el origen de la energía eléctrica y una carga que recibe esta energía y la convierte a otras formas de energía tales como calor, luz o trabajo mecánico. En general un circuito eléctrico o electrónico consta de una serie de ***fuentes*** (fuentes de poder) y ***elementos*** (dispositivos de carga) unidos por conectores. Los puntos de conexión son llamados nodos o terminales. Los elementos eléctricos y electrónicos de nuestro interés son resistencias, capacitancias, inductancias, diodos, diodos emisores de luz, transistores y fototransistores. Los circuitos que contienen elementos tales como transistores y diodos son conocidos generalmente como circuitos electrónicos y se les conoce como ***activos*** pues cambian su estado en respuesta a una señal externa a diferencia de las resistencias, capacitores e inductores que no tiene esa capacidad y se les llama ***pasivos***.

2.1 Resistencia

Si se utiliza un material que no sea tan buen conductor como el cobre y le ponemos dos terminales, una a cada extremo, hemos construido lo que se llama una ***resistencia*** (figura *2.1*). El cuerpo de una resistencia comúnmente se forma de carbón, en el cual los electrones libres son poco numerosos. La presencia de una resistencia en un circuito eléctrico modera el desplazamiento de electrones.

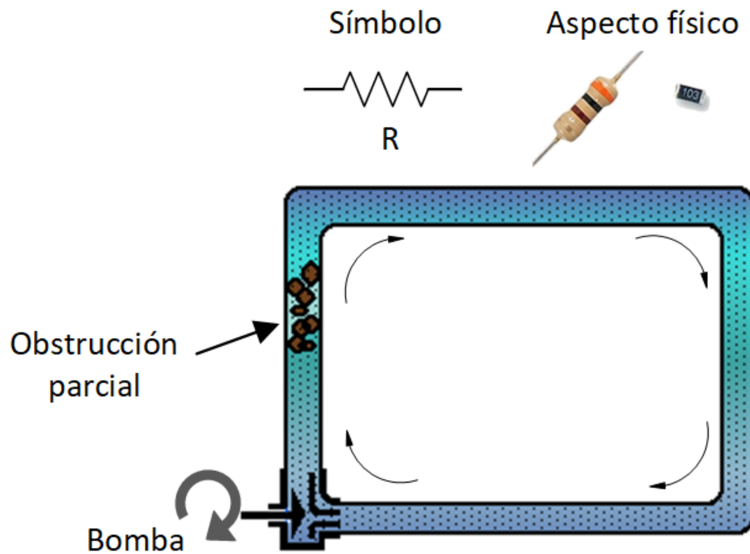


Figura 2.16 Resistencia: símbolo, aspecto y símil hidráulico.

Un símil hidráulico (figura *2.1*) sería llenar parcialmente de grava una tubería, en la cual se hace circular agua con una bomba. Es claro que la circulación del líquido se encuentra disminuida por este trozo parcialmente obstruido.

Otra comparación consiste en recordar el tráfico de los autos en una vía cualquiera; si la vía se estrecha por cualquier razón, la circulación se encuentra considerablemente obstruida, los coches deben circular en fila india, de forma que se producen largas filas de espera en la parte de libre circulación de coches, que es la parte de resistencia del circuito.

Las unidades de las resistencias son los ***Ohmios*** (Ohm, Ω) cuanto más se eleva este valor, mayor dificultad ofrece este elemento para el desplazamiento de electrones. En los esquemas se muestra como una línea quebrada y, próxima a ella, su valor en Ohmios (ver figura *2.1*). Físicamente la gran mayoría son similares a un pequeño cilindro con dos conductores saliendo de sus extremos; para su identificación tienen marcado el valor o unas bandas de colores que indican el valor de su resistencia. La última de las bandas indica la tolerancia en porcentaje ± de su valor teórico. En general un 5% o 10% de su valor.

El tamaño del cuerpo de la resistencia es proporcional a la potencia que se requiere disipar y dependerá de la corriente que circule por el elemento (ecuación *1.6*).

Los elementos de un circuito pueden ser conectados en serie o en paralelo (figura *2.2*) y es posible, en muchos casos, encontrar un circuito equivalente. En el caso de resistencias conectadas en forma paralela, podemos encontrar su valor equivalente aplicando las leyes de Kirchoff al circuito y tendremos:

(2.1) $$\frac{1}{R_t} = \frac{1}{R_1} + \frac{1}{R_2} + \cdots + \frac{1}{R_n}$$

Para el caso serial tendríamos:

(2.2) $$R_t = R_1 + R_2 + \cdots + R_n$$

donde $\boldsymbol{R_t}$ representa la resistencia total que el circuito ve.

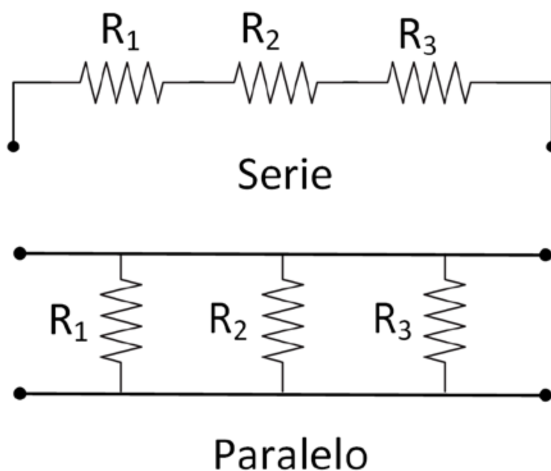
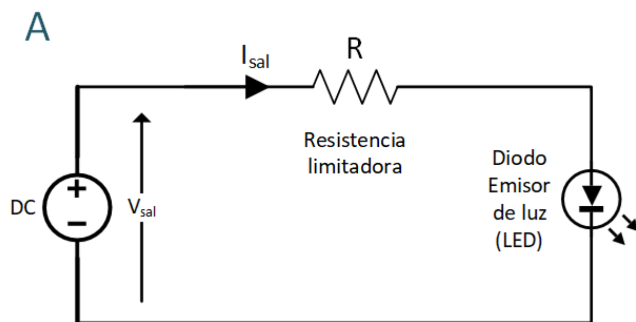


Figura 2.17 Resistencias conectadas en serie o en paralelo.

Una aplicación típica (figura *2.3*) es la de limitar la corriente que llega a otro elemento; un caso común es el del uso de una resistencia junto con un diodo emisor de luz (***LED***, ver sección de semiconductores) para evitar que el diodo se queme pues su resistencia interna es muy pequeña. Otro uso común es el de servir de unión entre una terminal de un interruptor que está conectada a tierra y la misma terminal conectada a un voltaje. Sin el uso de una resistencia tendríamos problemas al cerrar el interruptor pues toda la corriente tendería a irse por el camino de menor resistencia creando un corto circuito que dañaría a la fuente o al interruptor. Las resistencias son empleadas en calentadores y parrillas eléctricas. Al impedir el flujo de la corriente, se calientan, y si se calculan con cuidado pueden usarse para aprovechar la energía calorífica que desprenden.

Ciertas resistencias llamadas resistencias variables con la luz presentan cierta particularidad: su resistencia varía bajo el efecto de la luz incidente. Son llamadas ***LDR***.

Existen también resistencias cuyo valor se puede modificar a voluntad y son llamadas resistencias variables, potenciómetros o reóstatos. Una aplicación común es su uso para aumentar o disminuir el volumen de los radios. Su símbolo es el mismo que el de la resistencia fija cruzado con una flecha que indica la variabilidad.



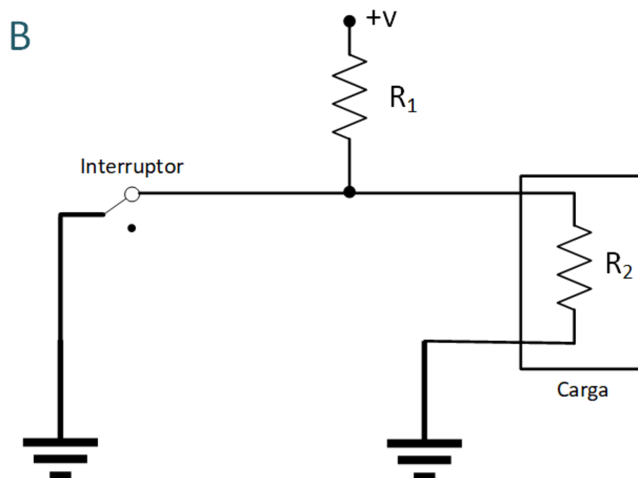


Figura 2.18 Resistencia usada como limitadora de corriente.

Ejercicio

2.1 Por medio de la experimentación se ha logrado determinar que la resistencia del cuerpo humano en condiciones desfavorables (con la piel húmeda) es de alrededor de 2,500Ω. Se sabe, también que el paso de una corriente de sólo 30mA a través del corazón es suficiente para que éste deje de latir. ¿Qué tensión se requiere, bajo esas condiciones para matar a un ser humano?

Respuesta
75V @ 30mA.

2.2 Para un circuito especifico se requiere en urgencia una resistencia de 1kΩ (bandas café, negra, roja al 5% de ¼ W). Buscando entre las disponibles se da cuenta que no tiene una de este valor, sin embargo, sí tiene varias resistencias de 2kΩ, 3kΩ y 6kΩ al 5% de ¼W ¿Qué puede hacer con las resistencias disponibles para obtener el valor requerido?

2.1.1 Midiendo Resistencias

El instrumento utilizado para medir la resistencia es el óhmetro[12]. El óhmetro (encuadrado generalmente en un multímetro analógico o digital) aplica, mediante una pila interna, una diferencia de potencial entre sus terminales, para luego, mediante un galvanómetro, medir la corriente que circula a través de la resistencia.

La escala del galvanómetro está calibrada directamente en ohmios. En aplicación de la ley de Ohm, al ser fijo el voltaje de la batería, la intensidad circulante a través del galvanómetro sólo va a depender del valor de la resistencia bajo medida, esto es, a menor resistencia mayor intensidad de corriente y viceversa.

Para medir una resistencia no olvide:

1. Usar la escala adecuada a la resistencia que se desea medir.
2. Verificar que el circuito no esté alimentado por ningún tipo de fuente so pena de alterar el valor de la lectura y/o dañar el instrumento de medición.
3. La medición en el circuito es siempre en ***paralelo*** con respecto al elemento a medir.

Si se requiere una mayor precisión en la medida se usa un puente de Wheatstone[13], que consiste en usar tres resistencias conocidas para averiguar el valor de una cuarta (la que nos interesa).

[12] inventado por el físico alemán George Simón Alfred Ohm.
[13] Desarrollado en 1833 por Samuel Hunter Christie. En 1843 Charles Wheatstone identifica uno de sus múltiples usos

2.2 Capacitor

En su forma más sencilla (ver figura *2.4*), un capacitor[14] o condensador está constituido de tres partes: dos placas metálicas, ***a*** y ***b***, conductoras llamadas ***armaduras***, y una delgada capa aislante[15] entre ellas llamada ***dieléctrico***.

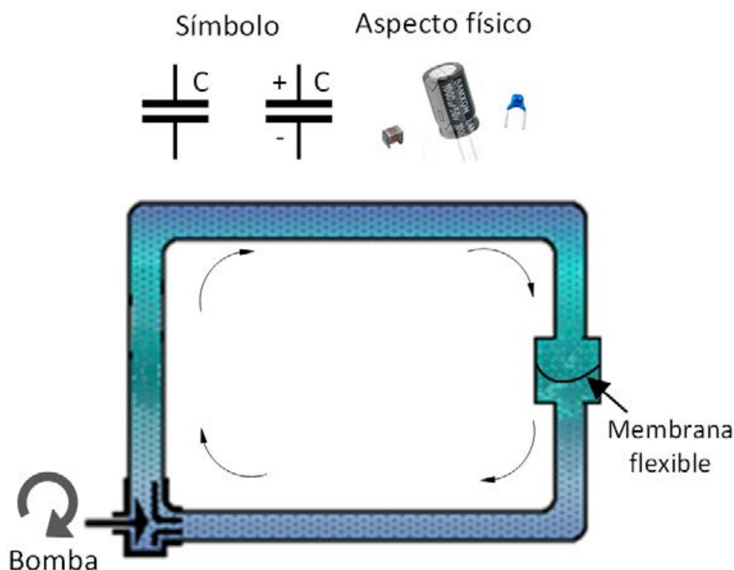


Figura 2.19 Capacitor: símbolo, apariencia física y símil hidráulico.

Suponga que una carga de ***q*** culombios se remueve de una armadura y se transfiere a la otra. Esto da por resultado una situación estable en que las cargas opuestas de las dos armaduras se atraen por medio de un campo eléctrico que pasa a través del dieléctrico; el dieléctrico evita que fluya una corriente entre las dos armaduras que neutralizaría su carga. En este estado se dice que el capacitor almacena una carga de ***q*** culombios. Si las terminales del capacitor ***a*** y ***b*** se conectan por medio de un conector externo, una corriente circula provocando que la armadura cargada del capacitor balancee su carga con la de la otra armadura. Puesto que es posible realizar trabajo con esta corriente, un capacitor es claramente un dispositivo que guarda energía en forma de cargas eléctricas en su armadura.

Si volvemos a nuestra tubería hidráulica (figura *2.4*) pero ahora interceptamos el flujo con una membrana elástica de caucho,

[14] La botella de Leyden, en Holanda, se considera como el 1er condensador. Es una botella de vidrio que almacena cargas que se consideraban se "condensaban" en su interior.
[15] Generalmente de mica, porcelana, aceite, aire o papel encerado (dependiendo la aplicación).

resulta que la bomba impulsa el agua para que circule, pero ésta no puede hacerlo debido a la membrana impermeable (en el caso real debería ser semipermeable).

Como la membrana es elástica se comba bajo la presión del líquido, dando por resultado un desplazamiento de toda la columna de agua hasta que la tensión mecánica de la membrana sea igual a la presión de la bomba. Esto es exactamente lo que pasa con los electrones. La pila impulsa a los electrones a circular, pero ninguno puede hacerlo porque nuestro dieléctrico (perfectamente aislante) no puede dejarles el paso libre. Pero la presión de la pila provoca una deformación de los átomos del aislante. Esta deformación corresponde a una modificación de la trayectoria de los electrones alrededor del núcleo. Es decir, el dieléctrico tiene una forma de elasticidad, comparable a la de la membrana. Esta elasticidad permite a los electrones del circuito, bajo la presión de la pila, desplazarse hacia adelante como las partículas de agua bajo el impulso de la bomba, hasta que la tensión de la pila equilibre la "deformación" de los átomos del aislante.

Este desplazamiento hacia adelante provoca un aumento de la densidad de los electrones sobre una de las armaduras del condensador y una disminución sobre la otra. Se dice entonces que el condensador está cargado.

Si el condensador guarda una carga de ***q*** culombios, existe una diferencia de potencial entre las placas metálicas de ***v*** voltios y la razón de la capacidad de carga ***q*** a la diferencia de potencial es una constante ***C*** llamada ***capacitancia*** y se expresa en faradios (***F***):

(2.3) $$C = \frac{q}{v} \; F$$

Cuanto más elevado es el valor de un capacitor, mayor es el desplazamiento de electrones bajo el efecto de la tensión de voltaje. Usualmente se emplean valores que van de los microfaradios (***μF***) a los picofaradios (***pF***). El esquema de un condensador se representa por dos trazas horizontales paralelas de igual longitud junto a las cuales se indica el valor del capacitor. Físicamente se encuentran de varias formas siendo la más común un pequeño cilindro o esfera con dos conectores en cuyo cuerpo se encuentra indicado el valor con caracteres o bandas de color.

Para encontrar el valor instantáneo de la carga ***q*** de un capacitor podemos relacionar la corriente ***i*** por unidad de tiempo:

(1.15) *(2.4)* $q = \int i\, dt$

por lo que también podemos escribir:

(2.5) $c = \frac{1}{v} \int i\, dt$

de donde:

(2.6) $i = c\, \frac{dv}{dt}$

Cuando un valor fijo externo se aplica a un capacitor ***C***, la carga se acumula rápidamente hasta almacenar ***q=Cv*** culombios. En este punto, ***C*** está cargado y no fluye más corriente. Así, un capacitor completamente cargado actúa como si fuera un circuito abierto. Si ***v*** es constante, entonces ***dv/dt=0*** e ***i=0***. Si el voltaje aplicado al capacitor es variable con el tiempo y de muy alta frecuencia (***dv/dt*** muy grande), el capacitor actúa entonces como un corto circuito.

Dado que solamente se requiere un período muy corto para cargar un capacitor con una fuente de voltaje de c.c., el flujo de corriente se detendrá casi instantáneamente. Por consiguiente, para un voltaje de ***CC***, el capacitor constituye un circuito abierto, pues evita el flujo continuo de corriente. Sin embargo, si un capacitor está conectado a una fuente de tensión alterna, se cargará primero en una dirección y luego en la otra, debido al constante cambio de polaridad del voltaje aplicado. Como resultado de esto habrá una continua inversión de corriente en el circuito, lo cual da la impresión de que la corriente alterna fluye por el capacitor. Y si bien esto no es lo que ocurre en realidad, el resultado es como si lo fuera. Por lo tanto, un capacitor permite el flujo de corriente alterna.

La cantidad de corriente alterna que fluye por el circuito de un capacitor depende de la oposición causada por el capacitor. Esta oposición se denomina ***reactancia***, que es otra forma de nombrar la resistencia al flujo de una corriente alterna, a causa de la capacitancia o inductancia (ver siguiente sección). Dado que en el caso que nos ocupa, la reactancia se debe a un capacitor, se llama ***reactancia capacitiva***, y el símbolo correspondiente es ***Xc***.

El valor de ***Xc*** depende del valor eléctrico del capacitor y de la frecuencia del voltaje aplicado. La relación matemática de estos factores es la siguiente:

(2.7) $$X_c \cong \frac{0.159}{FC}$$

en la cual ***Xc*** representa la reactancia capacitiva, ***F*** la frecuencia y ***C*** el valor de la capacidad en faradios. De acuerdo con esta fórmula, puede verse que a medida que aumenta la frecuencia, decrece la reactancia capacitiva correspondiente a una capacidad determinada. Estos principios tienen gran importancia en la radio comunicación.

El consumo de potencia de un capacitor se calcula de ***P=VI*** (*1.6*) y de (*2.6*):

(2.8) $$P = Cv\,\frac{dv}{dt}\;\;W$$

Puesto que la potencia es trabajo o energía generada por unidad de tiempo, la energía ***w*** guardada en ***C*** está dada por la relación:

(2.9) $$w = \int p\,dt\;\;J$$

ó

(2.10) $$w = \int Cv\,dt\; = \; {}^{1}\!/_{2}\; C\,v^2\;\;J$$

Los capacitores también pueden conectarse en serie o en paralelo. Si se conectan en serie:

(2.11) $$\frac{1}{C_t} = \frac{1}{C_1} + \frac{1}{C_2} + \cdots + \frac{1}{C_n}$$

si están en paralelo:

(2.12) $$C_t = C_1 + C_2 + \cdots + C_n$$

Al conectar un condensador cargado a un circuito, una resistencia en el caso más simple, éste inicia un proceso de descarga (ver figura *2.5*) cuyo tiempo dependerá de los valores ***R*** y ***C***. A esto se le denomina ***constante de tiempo*** que nos viene a decir cuánto tiempo tarda en descargarse el condensador a un 63.2% de su carga inicial.

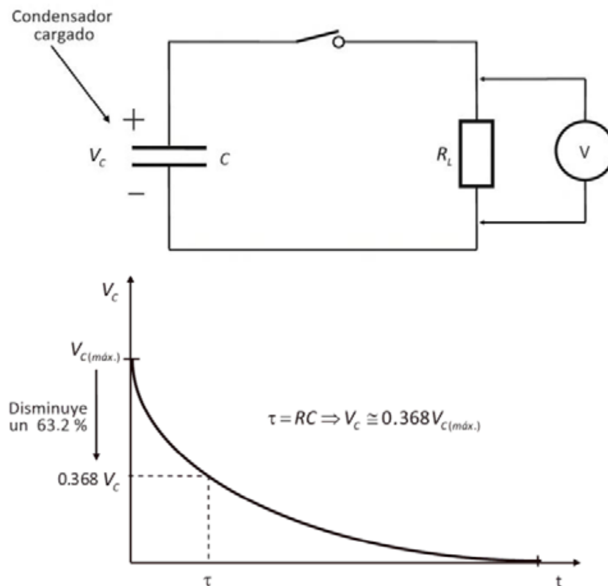


Figura 2.20 Constante de tiempo de descarga de un capacitor.

Al igual que en el caso de las resistencias, existen capacitores variables que cambian su valor por medio de un eje mecánico que mueve las armaduras acercándolas o alejándolas. Una de sus aplicaciones más comunes es su uso para variar las frecuencias que permite pasar un filtro formado por capacitores e inductores (ver siguiente sección) como, por ejemplo, el sintonizador de un radio. Su símbolo es el mismo que el del capacitor, pero con una flecha cruzándolo que indica la variabilidad.



Antes de intervenir en cualquier circuito eléctrico que cuente entre sus componentes con condensadores, éstos se deben descargar. Aunque muchos técnicos usan un sencillo destornillador para poner sus terminales en corto circuito, esto puede dañar el dieléctrico del capacitor. Es mejor usar una herramienta especialmente diseñada para descargarlo.

Un condensador puede explotar si se sobrepasa la tensión máxima especificada o si no se respeta la polaridad indicada (en el caso de los electrolíticos).

Al igual que las resistencias, los capacitores tienen una tolerancia expresada en un porcentaje con respecto a su valor teórico.

Idealmente, al aplicar una corriente a las armaduras de un capacitor, el dieléctrico debería actuar como un aislante perfecto e impedir el paso de la corriente una vez cargado el condensador. Sin

embargo, esto no es así. Existe una pequeña corriente que se “escapa” entre ambas armaduras y que se conoce precisamente como ***corriente de fuga***.

Si en los condensadores se usa un dieléctrico de óxido de tantalio o de aluminio se puede obtener una alta capacidad (en faradios). Estos condensadores, llamados ***electrolíticos***, tienen una polaridad que debe respetarse estrictamente, además de un voltaje máximo de operación.

Entre las aplicaciones comunes de los condensadores encontramos:

- Acoplamiento de señales entre etapas amplificadoras.
- Filtrado de circuitos de alimentación.
- Filtrado de señales de ruido eléctrico.
- Circuitos de filtrado en equipos de sonido.
- Eliminación de chispas en interruptores.
- Temporizadores.

2.2.1 Midiendo Capacitancias

La medición de la capacitancia se efectúa a través de un capacímetro, ya sea analógico o digital, que usualmente está integrado en un multímetro. El principio general de este instrumento de medición está en la reactancia capacitiva; es decir, en la medición de la corriente que circula en un condensador cuando se le aplica una tensión alterna de frecuencia fija, donde la corriente que circula dependerá exclusivamente de la capacitancia del componente o circuito que se prueba. Si el aparato de medida es más sofisticado podrá también medir la corriente de fuga, la resistencia del dieléctrico y la componente inductiva.

Antes de medir una capacitancia asegúrese de:

1. Usar la escala adecuada a la capacitancia que se desea medir.
2. Verificar que el circuito no esté alimentado por ningún tipo de fuente so pena de dañar el instrumento de medición.
3. La medición en el circuito es siempre en ***paralelo*** con respecto al elemento a medir.

2.3 Inductores

Hasta ahora sólo hemos considerado los fenómenos que surgen de los campos eléctricos que rodean a cada partícula cargada. Sólo hay campos de fuerza asociados con cargas estacionarias. Una carga eléctrica en movimiento también crea un campo magnético alrededor de sí misma. Así, un cable que lleva una corriente eléctrica también tiene un campo magnético asociado con propiedades muy similares a los de un magneto permanente.

Un ***inductor*** o bobina está constituido por un alambre de cobre aislado de más o menos longitud que se enrolla sobre sí mismo (ver figura *2.6*). Nada de particular debería ocurrir en el desplazamiento de los electrones en un circuito donde se ha insertado una bobina; los electrones, impulsados por la fuerza de la pila, deben desplazarse sin dificultad tanto si el hilo está enrollado como extendido. Éste sería el caso de no ser por el magnetismo que se genera alrededor de un cable que conduce corriente eléctrica.

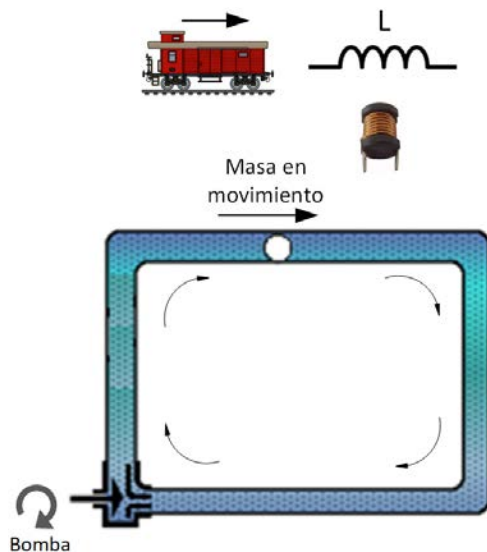


Figura 2.21 Inductor: símil, símbolo eléctrico y aspecto físico.

El comportamiento de un inductor es muy similar al de un capacitor que ya hemos explicado, pero aquí se usa el ejemplo de un carro de ferrocarril. Para poner en movimiento el pesado vagón, es necesario aplicar mucha fuerza que sólo se logra progresivamente y con un esfuerzo continuo. Es como si los electrones se hubiesen vuelto más pesados. Si se interrumpe la vía o se coloca un obstáculo, el vagón no se detiene de inmediato, sino que utilizará toda la energía cinética almacenada contra la obstrucción. En el caso de una bobina esto se traduce en un chispazo.

La inductancia se mide en Henrios. El valor de la inductancia de una bobina depende del número de espiras o vueltas del alambre. Cuanto más elevado el valor, más importante la masa aparente que esta autoinducción confiere a los electrones. Se utiliza corrientemente el milihenrio (***mH*** milésima parte de Henrio) y el microhenrio (***µH*** o millonésima parte de Henrio). En los esquemas se representa la inducción con una pequeña bobina junto con su valor. Físicamente se forma por un cilindro en el cual se embobina el conductor.



Joseph Henry
(1797-1878)

Científico norteamericano que descubre el fenómeno electromagnético de la autoinducción mientras construía electroimanes. Descubre, también, la inductancia mutua independientemente de Michel Faraday. Inventa el timbre eléctrico y el relé electromagnético e instaura las bases del telégrafo eléctrico inventado por separadamente por Samuel Morse y Charles Wheatstone. La unidad de inducción se mide en Henrios en su honor.

El voltaje a través de un inductor es directamente proporcional a la tasa de cambio de la corriente que circula por él:

$$v = L\,\frac{di}{dt} \tag{2.13}$$

El comportamiento de un inductor es muy parecido al de un capacitor con los voltajes y las corrientes intercambiados. Por ejemplo, la energía almacenada en los campos magnéticos es:

$$w = {}^{1}/_{2}\; L\, i^{2} \quad J \tag{2.14}$$

Cuando el arrollamiento de la bobina se realiza sobre otro material que no sea el aire, se le denomina de núcleo del material. Por ejemplo, si se enrolla sobre una barra de fierro se le llama bobina de núcleo de hierro y sus características son resaltadas (forma típica de crear un electroimán).

El uso típico de un inductor o bobina son múltiples: filtros (dispositivos que sólo dejan pasar algunas frecuencias), antenas, relevadores, contactores, bocinas, fonocaptores (usados en los tocadiscos), electroválvulas, timbres, motores eléctricos, frenos magnéticos, electroimanes y, de forma muy importante, transformadores y generadores.

Un ***transformador*** consiste en un par de arrollamientos de alambre de cobre aislado en un núcleo de hierro compartido dentro de una caja aislante de la que salen dos terminales de un inductor, por un extremo, y otras dos del otro inductor por el otro extremo. Su función consiste en aumentar o disminuir el voltaje de un lado u otro del transformador (dependiendo de la relación de vueltas de un inductor con respecto al otro) o para aislar un circuito de otro. El voltaje que sale de un lado obedece a:

$$(2.15) \qquad \frac{v_1}{N_1} = \frac{v_2}{N_2}$$

donde ***N*** es el número de vueltas de cada bobina.

Los relevadores eran un dispositivo muy usado en las primeras computadoras y en la telefonía antes de ser reemplazados con el transistor. Aún hoy en día se utilizan en algunas aplicaciones específicas, pero en la mayoría de los casos se usa su equivalente electrónico (***SSR***, ***TRIAC***). Un ***relevador*** o ***relé*** consiste en una bobina de núcleo de hierro por la que se hace circular una corriente. Al circular los electrones por la inductancia, se genera un campo magnético que el núcleo de hierro encausa hacia un interruptor que es accionado (o abierto) causando que otro circuito eléctrico se active; lo que permite realizar sistemas digitales donde existen dos estados identificados como abierto o cerrado.

Los inductores se encuentran también en una variedad que permite cambiar la inductancia a voluntad dentro de ciertos márgenes estrechos. Su aplicación principal es en el ajuste fino de filtros formados para eliminar o dejar pasar ciertas frecuencias de ondas de voltaje (por ejemplo, en la sintonización de un radio). Su símbolo es el mismo que la inductancia fija, pero con una flecha cruzándolo que indica que es posible variar el valor.

Es muy importante resaltar que todo elemento eléctrico o electrónico tiene, en mayor o menor grado, todas las características de resistencia, inductancia y capacitancia. Un conector de alambre no tiene resistencia cero, pero su resistencia es tan pequeña que muchas veces es despreciable. Puesto que hay cargas eléctricas en movimiento también existen campos magnéticos por lo que hay fenómenos de inductancia y capacitancia que en ciertos análisis es posible ignorar. Así, un alambre que lleva corriente puede interactuar con otros a su alrededor induciendo pequeñas corrientes no deseadas llamadas ***ruido***.

El ***generador*** eléctrico, en nuestro caso un ***alternador***, consiste en una máquina que transforma energía mecánica (de movimiento de giro) en energía eléctrica. Esto se basa en el principio de que cuando un conductor (en la práctica, una bobina) corta un campo magnético, se genera en el conductor una fuerza electromotriz (f.e.m. o ***FEM***). Su estructura es muy similar a la de un motor eléctrico (refiérase a la figura *1.6*).

En las centrales eléctricas se usan alternadores con una estructura interna que consta de tres bobinas o fases generando corriente alterna trifásica desfasada 120° entre sí. Al haber 3 bobinados a la salida del alternador aparecen 6 cables que se interconectan para distribuirse como 3 fases y un neutro (tierra) para, finalmente, después de pasar por un transformador, llegar a nuestra casa como una fase y un neutro (países con 120v) o dos fases y un neutro (países con 220v).

2.3.1 Midiendo la Inductancia

La inductancia se mide por procedimientos indirectos:

1. Con un medidor ***LCR*** que generalmente se incluye en un multímetro especializado. La medición se realiza en serie con la inductancia.
2. Utilizando una resistencia de precisión en serie con la inductancia además de un generador de frecuencias y un osciloscopio. Aplicar la fórmula:
$$L = \frac{R}{(2 \times \pi \times f \times \sqrt[2]{3})}$$
3. Calcular la inductancia sobre una pendiente de voltaje corriente con un generador de pulsos y un osciloscopio.
4. Con un condensador, una resistencia de precisión, un generador de señales y un osciloscopio.

Como se ve, el método más fácil y directo es el número uno.

Ejercicios

2.3 Generando ***CA***. Si tiene acceso a un multímetro y una bocina (altavoz), de preferencia con un cono grande, podrá realizar la siguiente experiencia: conecte el multímetro en la escala más sensible de corriente (50µA) en paralelo con el altavoz. Haga vibrar el cono con los dedos, suave, pero firmemente ¿Qué sucede? Explique.

2.4 Requerirá un clavo (o tornillo) de hierro y otro de aluminio de unos 10cm, cinta de adhesiva, dos metros de alambre eléctrico de cobre calibre 28, un lápiz y una pila tipo A de 1.5V. Enrolle el cable alrededor del clavo de hierro dejando alrededor de 30cm en cada extremo. Quite el aislante de los dos extremos del cable y aplique las terminales a la pila usando la cinta adhesiva asegurando un buen contacto. Acerque el dispositivo a pequeñas piezas metálicas. ¿Qué sucede? ¿Qué pasa si en lugar

de usar *núcleo* de hierro usa aluminio, madera o aire? (Para ello envuelva el alambre al clavo de aluminio o al lápiz) Explique.

2.4 Dispositivos Semiconductores

Con el descubrimiento de las propiedades de los semiconductores se hizo posible todo un desarrollo tecnológico que nos ha llevado a avances insospechados en la ciencia. Era común construir cada elemento por separado y luego unirlos por medio de conductores metálicos para formar un circuito funcional. Hoy en día se acostumbra a crear todos los componentes en un sólo cristal de silicio (***Si***) o silicio−germanio (***GeSi***) y empaquetarlos en un envase de plástico con conexiones externas metálicas, pero las inductancias aún no se han logrado del todo. En consecuencia, los tamaños y precios de los circuitos han bajado considerablemente, siendo ahora más rentable reemplazar que reparar.

Para comprender cómo los transistores y otros elementos pasivos pueden fabricarse en un cristal de silicio, es necesario considerar la naturaleza física de los materiales semiconductores. En un conductor, tal como el metal, la corriente es portada por electrones que están libres para moverse por la estructura atómica de la sustancia. En un aislante, todos los electrones están fuertemente ligados a las órbitas de sus átomos o moléculas y, por lo tanto, ninguno está disponible para servir de portador a las cargas eléctricas. La situación de un semiconductor es intermedia entre los dos: los portadores libres de cargas eléctricas no están disponibles usualmente, pero pueden ser generados con un modesto gasto de energía.

Un átomo de silicio (***Si***) tiene cuatro electrones en su valencia, u órbita de electrones más externa; en el silicio sólido, pares de electrones, compartidos por los átomos vecinos, están arreglados simétricamente de forma tal que cada átomo está rodeado de ocho electrones compartidos. Puesto que todos los electrones están ocupados en la liga entre los átomos, un cristal de silicio es un pobre conductor de la electricidad.

Los dispositivos semiconductores se logran introduciendo una cantidad controlada de ***impurezas*** en los cristales de silicio. A este proceso se le conoce como ***dopar***. Por ejemplo, parte de un cristal de silicio se puede contaminar (dopar) con arsénico (***As***), fósforo (***P***) o antimonio (***Sb***), elementos que tienen cinco electrones en su valencia. Un átomo de fósforo puede desplazar a un átomo de

silicio sin romper la estructura cristalina, pero el electrón extra que trae no tiene sitio en las ligas interatómicas. En la ausencia de estímulo externo, el electrón permanece en la vecindad del átomo de impureza (fósforo), pero puede ser movido aplicando un pequeño voltaje a través del cristal.

A un cristal contaminado así se le conoce como de ***tipo N***, porque tiene electrones (de carga negativa) que son predominantes o ***portadores mayoritarios*** de corriente eléctrica.

El silicio también puede ser contaminado con Boro (***B***), Galio (*Ga*) o Indio (***In***) elementos químicos cuyos átomos tienen tres electrones de valencia. Cada átomo de estos elementos insertado en la estructura del silicio crea una deficiencia de un electrón; un estado que es conocido como ***hueco***. Un hueco permanece asociado al átomo de la impureza en condiciones ordinarias, pero puede moverse aplicando voltaje al cristal. A un cristal contaminado de esta forma se le conoce como de ***tipo P***, puesto que los huecos cargados positivamente son los portadores mayoritarios, mientras los electrones son los ***portadores minoritarios***. Un hueco no es una partícula real, solamente la ausencia de un electrón en una posición en la que normalmente se encontraría en un cristal de silicio. De todas formas, el hueco tiene carga positiva y se mueve de forma muy similar a como lo haría una burbuja en un líquido. Un átomo adyacente transfiere un electrón al átomo de la impureza, “llenando” el hueco, pero creando uno nuevo en su propia nube de electrones; el proceso se repite, y el hueco va pasando de átomo en átomo.

2.4.1 Diodo

El dispositivo semiconductor más simple es el diodo (ver figura *2.7*) que se forma al unir un cristal de silicio tipo ***P*** con uno de tipo ***N*** (esto se hace en un sólo cristal). Cuando se aplica un voltaje positivo a la región ***P*** y un negativo a la ***N*** (polarización directa) se estable una contracorriente de electrones y huecos. Los huecos de la región ***P*** son repelidos por las cargas positivas aplicados a la terminal ***P*** y atraídos por la terminal negativa, así que fluyen a través de la unión. Los electrones de la región ***N*** son lanzados en la dirección contraria. La corriente que circula por el diodo se le llama ***corriente directa*** del diodo.

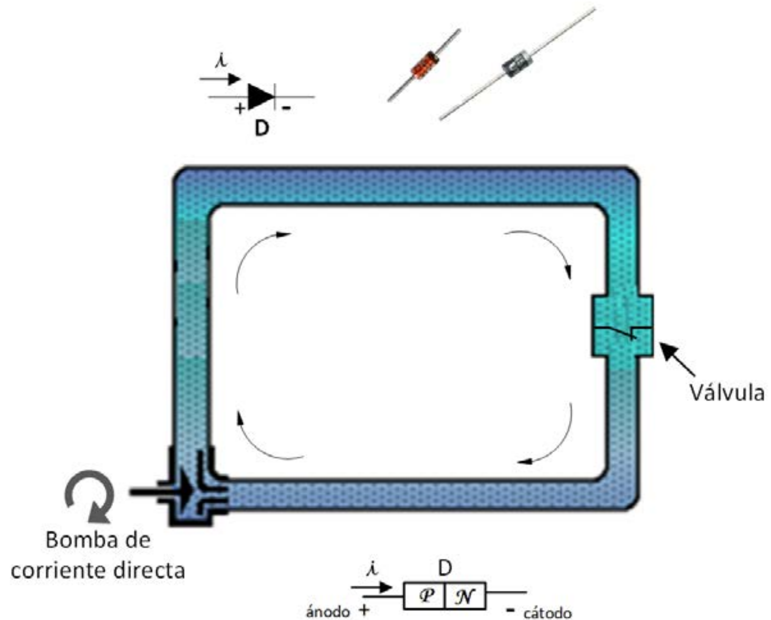


Figura 2.22 Diodo: símil, símbolo y aspecto físico.

Si las conexiones se invierten (polarización inversa), los huecos son retenidos en la región ***P,*** así como los electrones libres en la región ***N*** creando una gran zona en el centro en la que no hay ni huecos ni electrones libres por lo que es muy difícil que pase ningún tipo de portador. No fluye corriente por el diodo, excepto por una pequeña corriente llamada ***corriente inversa*** del diodo. Si seguimos aumentando el voltaje de polarización inversa del diodo llega un punto en que los portadores minoritarios, que circulan por el diodo creando una pequeña corriente, rompen la estructura de los cristales liberando nuevos portadores que a su vez liberan a otros en un efecto en cadena al que se le llama ***avalancha***, que, si no es controlado, destruye al diodo. Este efecto se utiliza en un tipo especial de diodos llamados ***Zener*** especialmente calculado para que, al llegar a la tensión para la cual fue calculado, el voltaje en sus terminales permanezca constante; aunque aumente la tensión de alimentación. Esta última característica lo hace sumamente útil en circuitos de alimentación y en la conversión de corriente alterna a directa regulando el voltaje de entrada (figura *2.8*). Su aspecto físico es similar al de un diodo común.

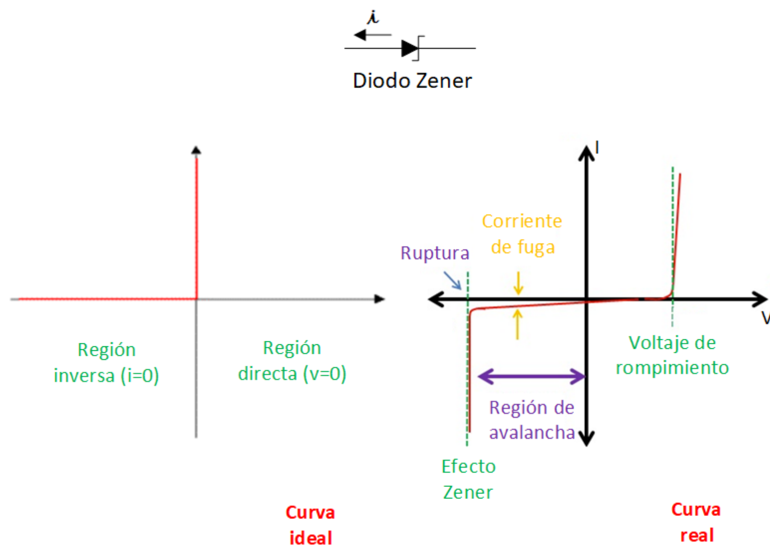


Figura 2.23 Diodo Zener y sus curvas.

El símil hidráulico de un diodo es una esclusa que sólo permite el paso del agua en una sola dirección. Un torniquete del metro guarda cierta similitud con este componente electrónico.

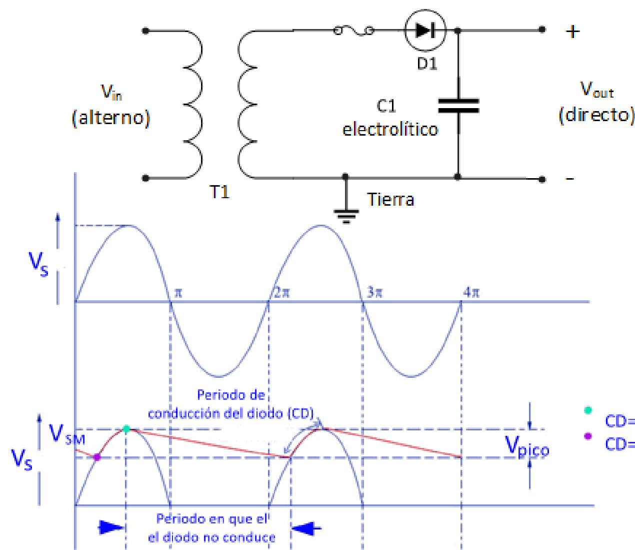
En los esquemas eléctricos se representa al diodo con una flecha y una línea perpendicular a está. La flecha indica hacia a dónde circula la corriente y siempre es de la región ***P*** (ánodo) a la ***N*** (cátodo).

La aplicación más común de un diodo es ***rectificar***[16] una onda de corriente alterna para convertirla en directa; junto con los capacitores y transformadores, forma el alma de casi toda fuente de alimentación de ***CC***.

Físicamente, son pequeñas ampollas de vidrio o cápsulas de plástico donde se indica con una flecha el sentido de circulación de la corriente o el cátodo con una banda. Existen de varios tamaños de acuerdo con la corriente que deben manejar.

Un arreglo común para rectificar una onda es el rectificador de media onda y el rectificador de onda completa que se muestran en la figura *2.9* junto con sus ondas resultantes.

[16] Corregir, enderezar.



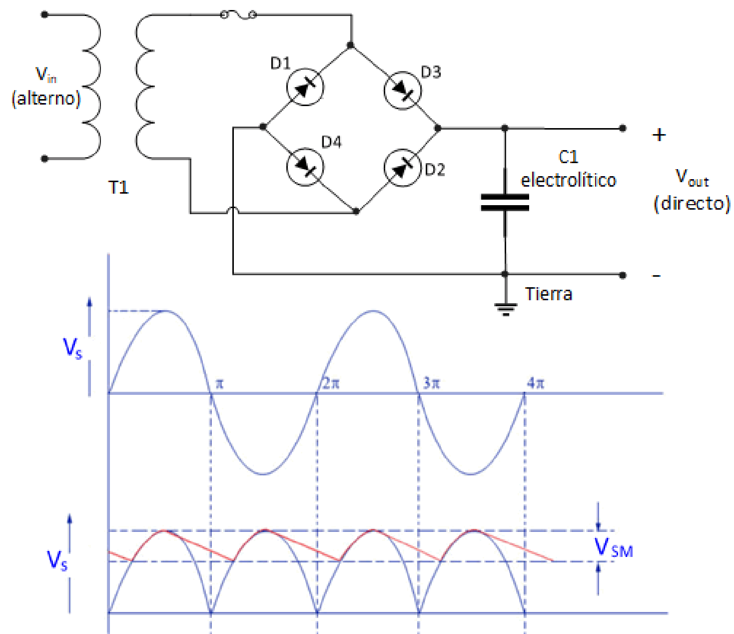


Figura 2.24 Rectificación de media onda y de onda completa.

Nótese que, en general, se debe reducir el voltaje de entrada para adaptarlo al de salida del circuito (5 o 12v ***CD*** en circuitos digitales)[17]. A la salida se incluye un capacitor electrolítico que ayuda a

[17] En general un circuito de este tipo no es suficientemente estable para usarse en circuitos digitales por lo que se prefiere el uso de fuentes digitales conmutadas o de circuitos integrados completos que cumplen esta función, tales como el 7805 (5V), 7812 (12V) y similares.

"suavizar" la onda cargándose en el periodo de conducción del diodo y descargándose en el que no conduce.

2.4.1.1 Tierra

Notará en la figura *2.9* que existe un símbolo especial que indica la tierra. El concepto de voltaje es relativo. Al decir que un punto en un circuito tiene un voltaje de 10V no significa nada si no se tiene otro punto del mismo circuito con el cual compararlo. En general se define un punto de referencia del circuito de 0 voltios sobre el cual todos los demás se basan. A este punto se le denomina ***tierra*** y, dependiendo el tipo de tierra, se le representa con cualquiera de los símbolos de la figura *2.10*.

Una correcta selección de la tierra puede decidir si el voltaje de un punto a otro es positivo o negativo con respecto a la tierra del circuito. Esto puede presentar confusiones al principio.

Si la tierra es una conexión física formada por una o varias varillas de cobre o aluminio insertadas en el suelo a una profundidad de 3 metros o más, se habla entonces de una ***tierra física*** o ***masa***. Esta "tierra" forma un sumidero infinito de electrones por lo que, prácticamente, se define como de potencial cero. Al conectarse todos los circuitos eléctricos a esta tierra común, todos ellos comparten el potencial de referencia de la tierra y, por lo tanto, un punto de referencia común.

La conexión a la tierra física de un instrumento se logra a través de un cable de alimentación que internamente se conecta a su chasis metálico por medio de tornillos o soldado. Es, por lo tanto, de suma importancia no cortar el sistema de tierra del conector o usar un adaptador que lo inhabilite. Si una de las tierras de los aparatos que se conectan a un sistema común no comparte la misma tierra se dice que está ***flotando***.

Algunas de las ventajas de no tener "tierras flotantes" son:

- Se disminuye el riesgo de electrocución en caso de falla o corto circuito del instrumento puesto que la ruta de flujo de la corriente en ese caso sería a través del chasis hacia la tierra física y no hacia el cuerpo del usuario.
- "Aterrizar" correctamente un equipo evita dañarlo con corrientes electrostáticas de descarga (***ESD***) generadas cuando un cuerpo cargado entra en contacto con equipo

delicado. Este cuerpo cargado puede ser usted, después de caminar por una alfombra sintética.

La causa más común del ruido eléctrico en sistemas electrónicos se debe a una mala práctica de unión de sistema de tierra. Si se usan varios puntos para la conexión a tierra se pueden causar diferencias de potencial que crean ***bucles de tierra*** que, a su vez, causan errores en las lecturas de voltaje. Una forma simple de remediar esto es conectar todo a un punto único de tierra. En práctica esto es difícil de implementar por lo que generalmente se usa un bus de tierras o barra de tierra común.

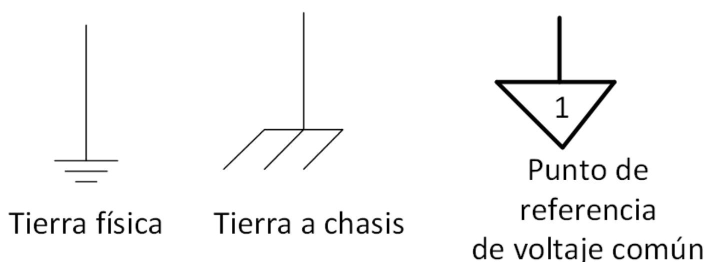


Figura 2.25 Símbolos de tierra.

2.4.2 Diodo Emisor de Luz (LED)

El diodo se utiliza también en casi todo componente electrónico moderno como indicador o despliegue de números o caracteres. De esta forma el diodo se modifica para que, al pasar una corriente eléctrica por él, emita luz. El diodo emisor de luz es comúnmente conocido por sus siglas en inglés: ***LED*** (Light-Emitting Diode). Todo ***LED*** debe acompañarse por una resistencia que limita el paso de la corriente. Un ***LED*** emite una luz proporcional a la corriente que conduce, por lo que la resistencia debe calcularse cuidadosamente para obtener un brillo máximo sin quemar el componente (ver figura *2.11*).

Otra aplicación del ***LED*** es como acoplador óptico (optoacoplador), esto es, la unión entre dos circuitos diferentes por medio de la luz, de forma similar a como lo hacía un relevador magnético. En este caso se usa un ***LED*** y un fototransistor.

Los ***LEDs*** se acomodan muchas veces en matrices para poder formar números o letras y es común verlos en todo tipo de despliegues económicos. Recientemente los ***LEDs*** de alta potencia (1 mm

cuadrado) sustituyen con éxito a los focos (bombillas) caseros, industriales y de la industria automóvil[18].

Esta substitución trae consigo varias ventajas. Podemos enumerar entre ellas:

- Es durable.
- Tiene un bajo consumo de corriente por el mismo número de lúmenes[19] (casi un 80% menos).
- Casi no se calienta.
- Su producción en serie es extremadamente barata.

Entre sus desventajas encontramos que el ***LED*** es caro en comparación con la antigua tecnología, su luz es lineal y baja de rendimiento a altas temperaturas.

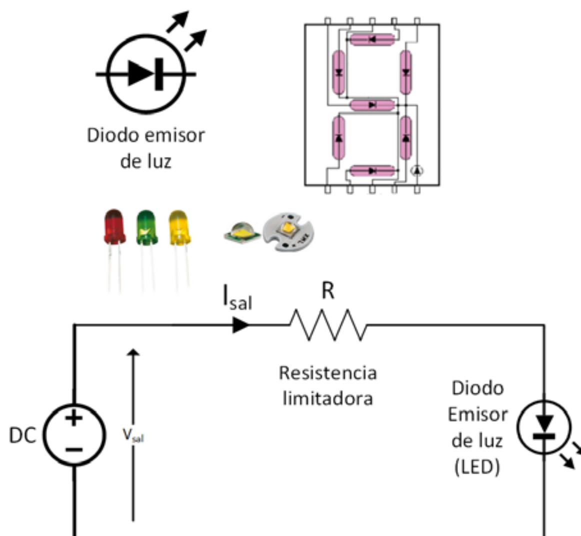


Figura 2.26 Diodo emisor de luz: símbolo, despliegue de 7 segmentos, aspecto físico y conexión eléctrica.

Un diodo no es capaz de ganancia, por lo que no es posible usarlo como un elemento activo; sin embargo, tiene una propiedad que lo distingue de los otros elementos pasivos. Las resistencias, capacitancias e inductancias son simétricas: sus efectos en la señal son los mismos no importando la polaridad de la señal y la forma en que se conecten al circuito. La característica más notable de un diodo es su asimetría: presenta una baja resistencia a una señal

[18] Su color no es blanco, sino que se obtiene combinando el rojo, verde y amarillo (RGB) o RYGB: rojo, amarillo, verde y azul. En otros casos es azul contaminado con fosforo para obtener blanco.

[19] Potencia emitida en forma de radiación luminosa a la que el ojo humano es sensible.

en un sentido de su polaridad y muy alta resistencia en el sentido contrario.

Ejercicio

2.5 Observe la figura *2.11* y calcule la resistencia necesaria si se sabe que el voltaje de la fuente es de 5 volts; la corriente necesaria para encender el ***LED*** rojo es de 20 miliamperios y existe una caída de voltaje de 2 Voltios en el diodo emisor de luz (valores típicos).

Respuesta:
150Ω.

2.4.3 Transistor

Un transistor se forma agregando una tercera región contaminada a un diodo, de forma tal que quede un "sándwich" de un cristal tipo ***P*** entre dos tipos ***N*** (transistor tipo ***PNP***, o un cristal tipo ***N*** entre dos tipos ***P***, transistor tipo ***NPN***). Una de las áreas ***N*** se conoce como el ***colector*** y otra como el ***emisor***; la región ***P*** entre ellas se llama ***base***. En su estructura, al transistor se le puede considerar como dos diodos, uno a espaldas del otro; en un mismo cristal de silicio. Como puede esperarse de este análisis, la operación del transistor depende de los voltajes relativos aplicados en cada una de las tres regiones que lo forman (ver figura *2.12a*).

Si vemos el símil hidráulico de la figura *2.12 b*, observamos un recipiente al que dos tabiques dividen en tres compartimentos, a los que llamaremos ***E***, ***B*** y ***C*** (de base, colector y emisor). En la base del tabique, entre ***B*** y ***E*** se encuentra una compuerta (válvula) D_1 que se abre solamente de ***B*** hacia ***E***. En la base del tabique entre ***C*** y ***B*** se encuentra una compuerta: D_2; puede subir y bajar gracias a un dispositivo mecánico, controlado por D_1. Una bomba, P_1, comunica los compartimientos ***B*** y ***E***. No gira más que en un solo sentido, aspirando el agua del compartimiento ***E*** e introduciéndola en ***B***. Una bomba, P_2, comunica el compartimiento ***C*** con ***E***; no gira nada más que en un solo sentido, aspira el agua del compartimiento ***E*** y la introduce en el compartimiento ***C*** (ver figura *2.12b*). Este conjunto es el equivalente hidráulico de un transistor. Los tres elementos del transistor, emisor, base y colector, corresponden a los tres compartimentos: ***E***, ***B*** y ***C***.

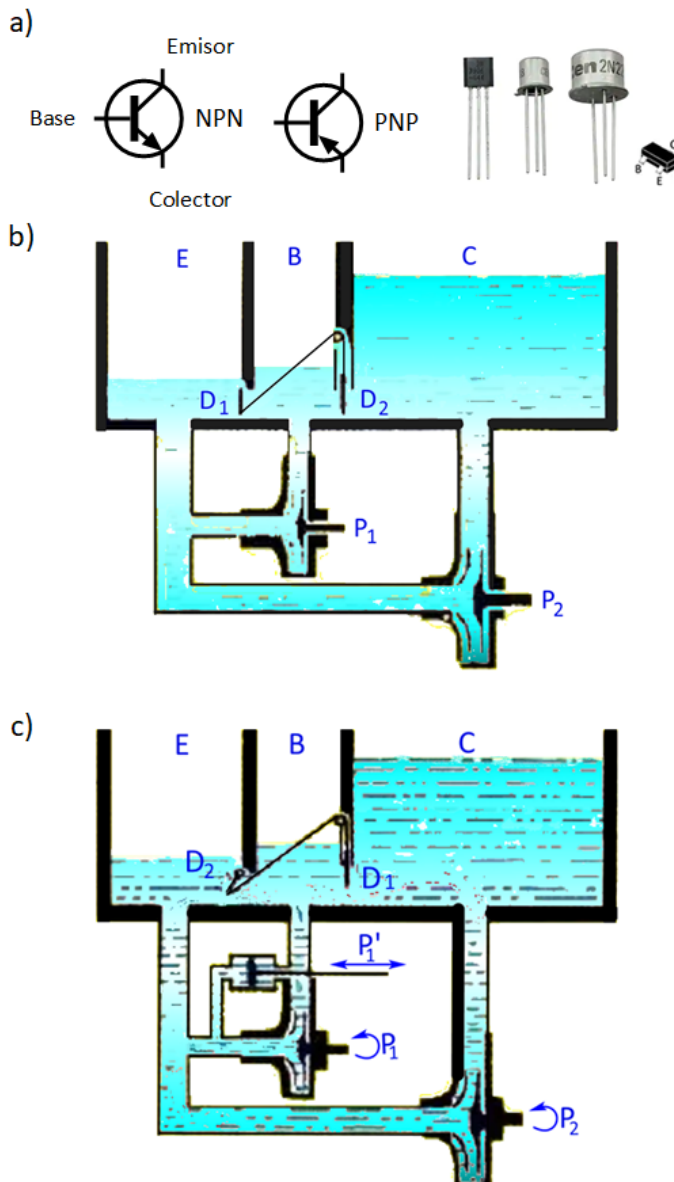


Figura 2.27 Transistor: símbolo, aspecto físico y símil hidráulico.

El emisor y la base de un transistor constituyen un diodo, al cual corresponde la compuerta $\boldsymbol{D_1}$. El colector y la base de un transistor constituyen otro diodo, comportándose en los montajes como lo hace la compuerta $\boldsymbol{D_2}$. Para poder funcionar, un transistor requiere de una corriente llamada de ***polarización*** entre la base y el emisor o la base y el colector (dependiendo si es ***PNP*** o ***NPN***). Esto corresponde a la bomba $\boldsymbol{P_1}$, que está conectada entre base y emisor del transistor. Otra fuente de corriente, a la cual corresponde

la bomba ***P_2***, está conectada entre colector y emisor del transistor. Estudiemos ahora el mecanismo del recipiente.

Cuando ***P_1*** no gira, ***D_1*** está cerrado al igual que ***D_2***. Si ***P_2*** gira, ninguna circulación se produce en el recipiente, porque hay dos tabiques herméticos entre los compartimentos ***C*** y ***E***. Cuando ***P_1*** gira, aspira el agua de ***E*** y la introduce hacia ***B*** cambiando todo el sistema, pues ahora la presión del líquido abre la válvula ***D_1***. Entonces la compuerta ***D_2*** se eleva y ***P_2*** hace circular el líquido de ***C*** hacia ***B*** y después hacia ***E***. La corriente que circula entre ***B*** y ***E*** bajo la acción de la bomba ***P_1***, es siempre una corriente débil porque no hay gran esfuerzo que desarrollar para levantar la compuerta ***D_2***. La corriente que circula entre ***C*** y ***E*** bajo la acción de la bomba ***P_2***, es siempre una corriente intensa porque la compuerta abre la entrada a una gran circulación de agua.

Se puede pues, controlar una gran corriente por medio de una pequeña bomba: ***P_1*** es una bomba de mando, pues ella abre la compuerta ***D_2***; ***P_2*** es una bomba de alimentación porque hace circular el agua. Esto mismo sucede en un transistor: La corriente intensa que circula entre colector y emisor bajo la acción de la pila ***P_2***, es controlada por una corriente débil que circula entre base y el emisor, bajo la acción de la pila ***P_1***. ***P_1*** es llamado voltaje de ***polarización*** y ***P_2*** voltaje de ***alimentación***.

El transistor permite dirigir "grandes cosas" con pequeños medios.

Los transistores ***NPN*** y ***PNP*** forman una familia de transistores llamados transistores de ***juntura***. Se les conoce también con el nombre de ***bipolares*** porque hay portadores de las dos polaridades envueltos en su funcionamiento. El transistor bipolar fue el primero en inventarse por John Barbeen, Walter H. Brattain y Willima Shockley de los laboratorios telefónicos Bell en 1948.

Existe un segundo tipo de transistor conocido como ***MOS*** (semiconductor de óxido de metal) o transistor de efecto de campo (FET). Aunque fue descubierto 25 años antes que el transistor, su fabricación no fue posible hasta los años sesenta. Tal como el transistor bipolar existe en dos formas complementarias ***pMOS*** y ***nMOS***, donde el prefijo ***p*** o ***n*** indica la polaridad de los portadores de carga mayoritarios usados para conducir corriente en este tipo de transistor y un tercer tipo que contiene a las dos anteriores llamado ***CMOS*** (Complementary Metal Oxide Semiconductor). No

analizaremos aquí este tipo de transistor y al lector interesado lo referimos a libros de la bibliografía.

2.4.3.1 Transistor como Amplificador

Por la capacidad de controlar grandes corrientes con pequeñas corrientes de control, un transistor puede ser usado como amplificador. Para entender su funcionamiento refirámonos una vez más a la figura *2.12c*.

Coloquemos una pequeña bomba alterna en el circuito de P_1. La llamaremos P_1'. Su pistón se mueve tanto en un sentido como en el otro. La bomba refuerza y disminuye la acción de P_1 alternativamente. La válvula D_1 sigue todas estas variaciones porque la corriente que lo eleva unas veces aumenta y otras veces disminuye. La compuerta D_2 se comporta exactamente igual: sube y baja alternativamente.

La corriente que va del compartimiento ***C*** hacia ***E*** bajo la presión de P_2 aumenta también cuando D_2 se eleva y disminuye cuando D_2 baja. La bomba P_1' gracias al mecanismo, actúa sobre un circuito en el cual no está directamente colocada. Ella hace aumentar o disminuir la corriente debida a P_2, de una forma mucho más importante colocada en el circuito de P_2. Decimos "mucho más importante" porque las débiles variaciones de corriente de mando se traducen, gracias a este mecanismo, en variaciones semejantes, pero amplificadas debido a la corriente de P_2.

Esto es exactamente lo mismo que ocurre en un transistor. Se pueden obtener variaciones muy importantes de corriente debidas a P_2, partiendo de variaciones semejantes, pero mucho más débiles que una corriente debida a P_1. Para hacer esto se utiliza una bomba alternativa P_1' que se coloca en el circuito de P_1. Este montaje es llamado ***amplificador*** y el trabajo que se efectúa de esta forma se le conoce como amplificación (ver figura *2.13*).

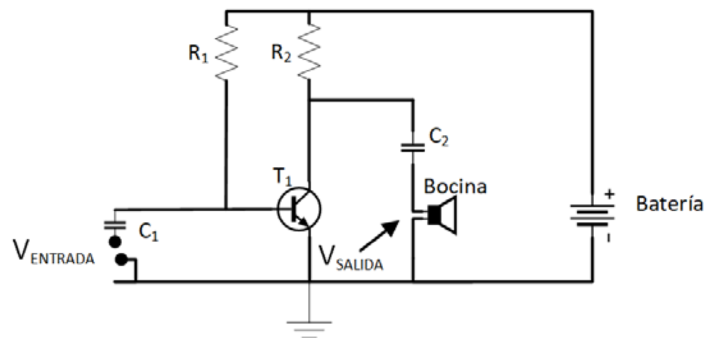
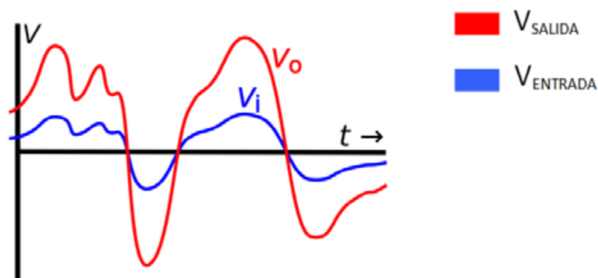


Figura 2.28 Circuito amplificador.



2.4.3.2 Comportamiento del Transistor

El transistor se usó en un principio como amplificador substituyendo rápidamente a los bulbos para la mayoría de las aplicaciones[20]. Las características de un transistor son generalmente estudiadas conectando un transistor a un circuito, variando las corrientes y voltajes de entrada y realizando la gráfica correspondiente de las corrientes y voltajes resultantes. En la figura *2.14* se muestran las corrientes y voltajes de un transistor y se observan en sus curvas tres regiones bien definidas:

1. ***Activa***. Se conoce como región activa cuando la corriente que fluye a la base, I_b, es positiva y V_{ce} es más positiva que V_{be} (el colector más positivo que el colector). Es en esta región donde se realiza la amplificación y es una zona aproximadamente lineal.

2. ***Corte***. El emisor y colector son negativos con respecto a la base. Como resultado, tanto el emisor como el colector aparecen como una resistencia grande al flujo de corriente. Sin embargo, alguna corriente fluye en sentido contrario pues hay una fuga por la junta (corriente de corte).

[20] Aún se usan en amplificadores de audio de alta potencia de calidad.

3. ***Saturación***. El emisor y colector están polarizados en directa (positivos) con respecto a la base, y tanto la juntura colector-base como la emisor-base presentan poca resistencia al flujo de la corriente.

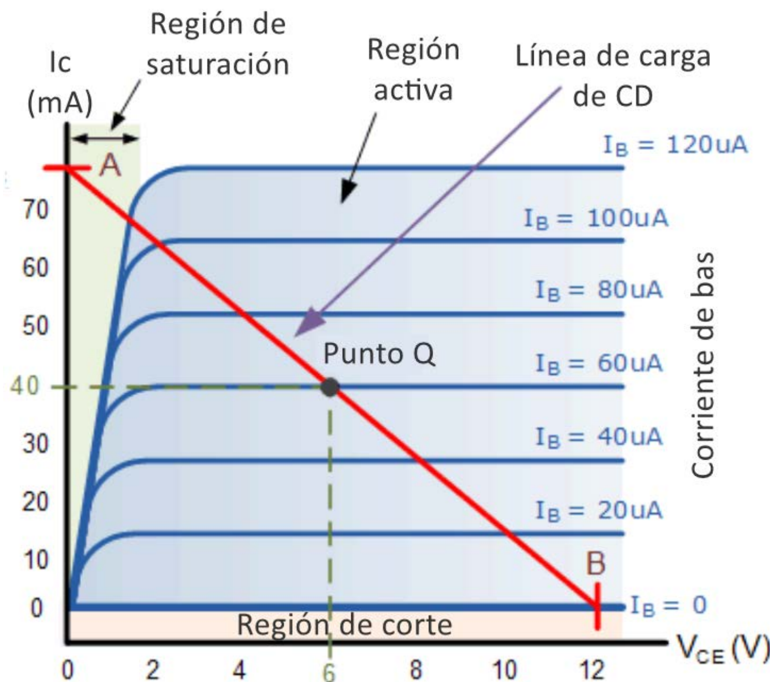


Figura 2.29 Curvas de un transistor.

La región lineal, sumamente importante en otros ámbitos, no nos interesa en nuestro estudio de circuitos digitales. La región de saturación y corte, por otro lado, son de sumo interés en este libro pues es la región utilizada para todo circuito digital (de dos estados posibles). En esa zona el transistor o conduce corriente o no, lo que muchas veces se conoce como "0" o "1" y un sólo transistor funciona como ***inversor*** (***NOT***, nuestro primer elemento digital), esto es, si tenemos corriente en la base, no tenemos salida en el colector y si no tenemos corriente en la base, tenemos salida en el colector (figura *2.15*).

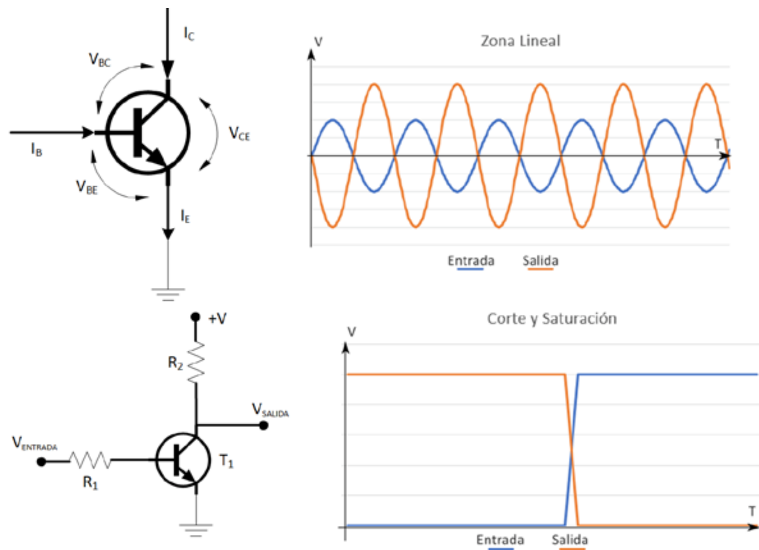


Figura 2.30 El transistor usado como inversor.

Note que:

- Se selecciona en qué zona funciona el transistor escogiendo cuidadosamente los valores de las resistencias R_1 y R_2.
- El transistor funciona como amplificador en su zona lineal.
- El transistor funciona como inversor en su zona de corte y saturación.

Cuando se usa el transistor como amplificador (región activa) se sigue el ***modelo de Ebers-Moll*** que expresa:

(2.16) $$i_C = \beta i_B$$

donde a β se le conoce como la ***ganancia*** del transistor y normalmente va de 10 a 100.

Y si aplicamos las leyes de Kirchoff al transistor tenemos:

(2.17) $$i_E = i_B + i_C$$

Recuerde que el uso del transistor como amplificador es muy importante, pero a lo largo de este libro sólo nos interesaremos en esa característica ocasionalmente. Las regiones de corte y saturación son las de más interés para nosotros en el diseño digital.

2.5 Circuitos Integrados (CI)

La electrónica moderna hace uso extensivo de los circuitos integrados donde, tanto componentes pasivos (resistencias, capacitores, inductores, diodos) como activos (transistores), se fabrican en un sólo substrato de silicio (chip) llamándolos ***circuitos integrados*** (***CI*** o ***IC*** en inglés). En esta sección analizaremos los métodos empleados para fabricar estos ***CI*** y cada uno de los elementos que los forman.

2.5.1 El Proceso de Manufactura

Los circuitos eléctricos se forman de conductores, aislantes y semiconductores combinados de forma funcional. Para fabricar un ***CI*** es necesario que se puedan combinar todos los elementos en una sola pastilla de silicio. El silicio es un elemento capaz de actuar como conductor, aislante o semiconductor, al modificarse sus características químicas y físicas por medio de contaminación o combinación con otros elementos químicos. La fabricación comienza con un conjunto de discos delgados de cristal de silicio llamados ***obleas*** o wafers, cada uno de ellos sirve como base o ***sustrato*** en el que una gran cantidad de ***CI*** idénticos son fabricados simultáneamente. La siguiente tabla indica en términos generales cómo un substrato es modificado para cumplir con propiedades eléctricas.

Propiedad de conducción	Material Usado	Método de fabricación
Conductor	Aluminio o polisílice	Depósitos en vacío
Semiconductor Puro	Silicio puro	Crecimiento de cristales de líquido
Semiconductor tipo P o N	Silicio contaminado	Difusión, implantación de iones, epitelial
Aislante	Bióxido de silicio o juntura PN	Oxidación o contaminación (ver arriba)

Las propiedades conductoras se obtienen agrupando cristales de silicio de forma irregular (polisílice) pero es más común el uso del aluminio como conductor. Se conocen varias técnicas para contaminar el silicio y así crear regiones del tipo ***P*** o ***N***. Un método muy popular es el de difusión, donde el substrato es calentado a altas temperaturas y expuesto a una atmósfera rica en el contaminante. El contaminante se difunde en la superficie para crear un material contaminado semiconductor (***N*** o ***P***). Los átomos de impurezas pueden también ser inyectados al silicio por medio de un

bombardeo de átomos llamado ***implantación de iones***. Un tercer método, llamado epitelial, consiste en depositar una delgada capa de átomos de silicio contaminado sobre la superficie expuesta del substrato.

Una propiedad muy importante del silicio es su baja conductividad al oxidarse (exposición al oxígeno). Esto se aprovecha para realizar aislantes oxidando la superficie del substrato de silicio para producir bióxido de silicio. Un segundo método de aislar consiste en colocar regiones ***N*** y ***P*** polarizadas en sentido contrario formando un diodo en inversa que actúa como un circuito abierto.

El proceso de manufactura descrito se caracteriza por crear nuevas capas cada vez que se requiere un nuevo elemento conductor, aislante o semiconductor. El espesor de estas capas se mide en micrómetros (10^{-6} metros) mientras que el área de cada ***CI*** no ocupa más de unos milímetros cuadrados. Un circuito integrado es pues un dispositivo de dos dimensiones o planos donde todos sus elementos están dispuestos de forma que no se traslapen, más que de forma limitada, en sus conexiones.

La construcción se realiza en capas en las que se crea cada una de un material específico en la superficie del substrato. Cada paso requiere que ciertas regiones sean expuestas y otras no. Esto se logra cubriendo las áreas que no se desean tratar. Para cubrirlas se usan patrones de recubrimiento fotográfico llamados ***mascarillas*** que siguen un ***patrón*** predeterminado y usando métodos análogos a los de fotograbación. La complejidad y costo de manufactura de un ***CI*** puede ser medido por el número de mascarillas (i.e. el número de operaciones de patrones) usadas, siendo seis un número típico. Un procesador moderno usa alrededor de 20 o más capas de substratos y hasta 6 o más para las interconexiones metálicas entre ellas.

Los últimos pasos en la elaboración son cortar cada uno de los ***CI*** de la oblea, probarlos de manera individual por medio de una computadora, colocar los conectores externos y encapsularlos en un empaque protector de plástico o cerámica. La forma de empaquetado más común, hasta hace pocos años era el ***DIP*** (empaquetado con conectores en paralelo) o ***SIP*** (empaquetado con conectores en serie). Debido a la gran cantidad de conectores externos

de algunos circuitos y a la miniaturización requerida[21], ambos casi han desaparecido en su uso industrial por otra forma de empaquetado llamada planar donde se prescinde de todo material plástico y conexión que no sea indispensable, reduciendo el área de un circuito de varios centímetros cuadrados a unos pocos milímetros cuadrados lo que da una mayor densidad de circuitos en un área determinada.

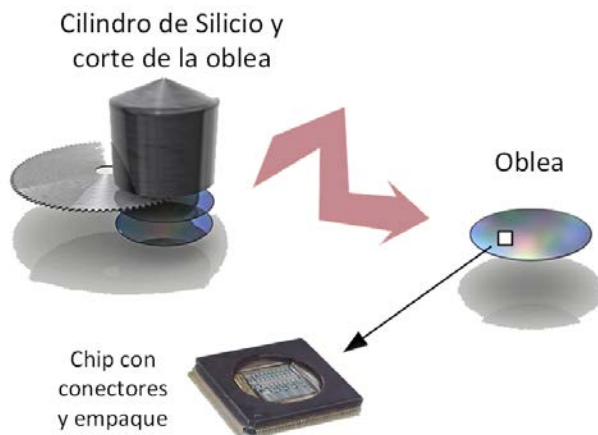




Figura 2.31 Fabricación de un circuito integrado (IC).

[21] El µprocesador Intel 4044 de 1era generación tenía 16 patillas de interconexión y 3,000 transistores, mientras que en 2020 el procesador de 10ma generación Core-i9 de Intel cuenta con 1151 conexiones (ya no se usan patillas sino almohadillas de cobre) y 2,600 millones de transistores.

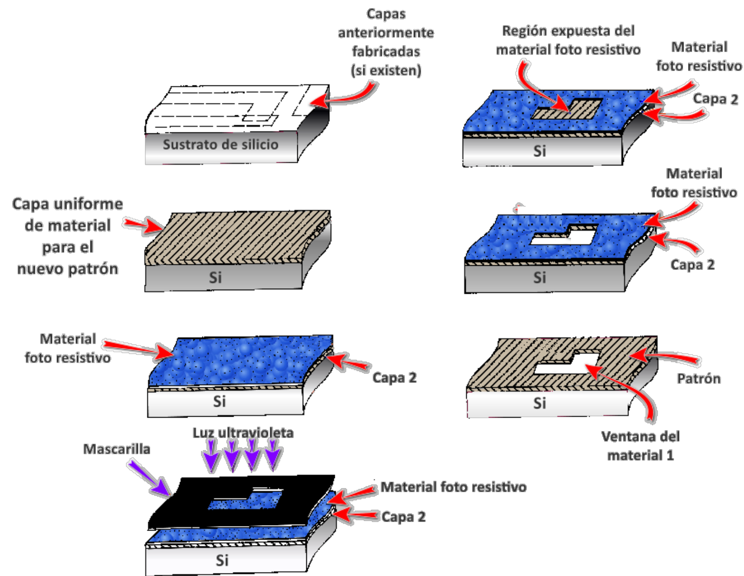


Figura 2.32 Fabricación de un circuito integrado (IC).

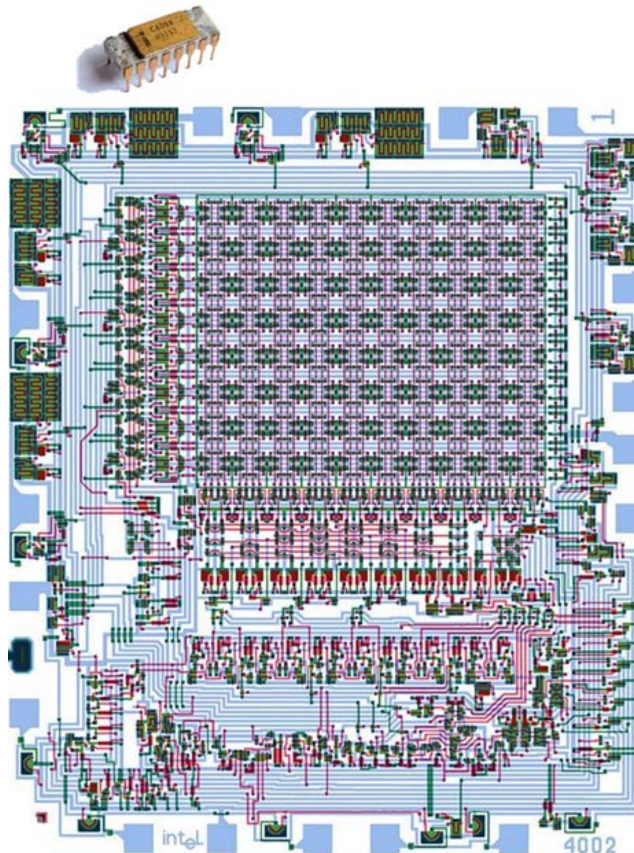


Figura 2.33 Microcomputadora 4004 de Intel, año 1971: 2,250 transistores. AMD Epyc Rome, año 2020: 39,540 millones de transistores.

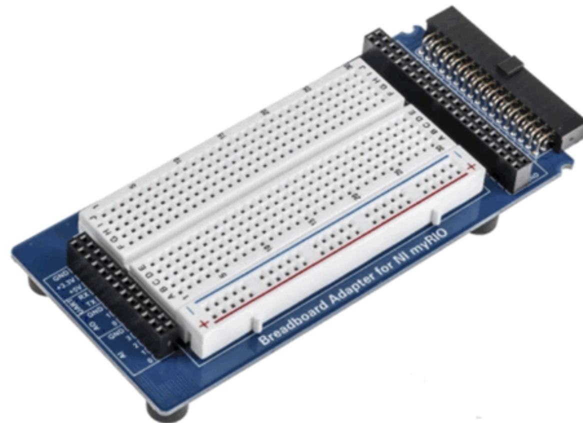
2.5.2 Métodos de Interconexión

Un sistema electrónico se forma de varios componentes y ***CI*** montados en una base adecuada e interconectados o alambrados juntos. Actualmente se usa como base una placa de fibra de vidrio, baquelita o plástico que provee soporte mecánico al sistema. Se usan varios esquemas de interconexión eléctrica entre los elementos, siendo el más común el circuito impreso.

El circuito impreso consiste en dibujar pistas de interconexión en una base de baquelita cubierta de metal (cobre); fijar por medio de un proceso fotográfico y eliminar el metal sobrante con un ácido que respeta las pistas. Se procede luego a perforar la placa de baquelita en los sitios donde se colocarán los componentes, colocar los componentes y soldar a las pistas. Las tarjetas de circuitos impresos procesadas de esta manera se usan extensivamente en la industria como un estándar. Pueden tener dos o más capas de interconexión en la misma tarjeta y son muy comunes las de dos (una de cada lado de la placa) que permiten las conexiones horizontales en un lado y las verticales en el otro evitando los cruces que no son posibles con este tipo de placas.

Otra forma de conexión es el uso de alambre cubierto de aislante que se enreda en bases especiales que soportan a los componentes. Usualmente se usa para producciones de bajas cantidades o con fines experimentales cambiando a circuito impreso para el producto final.

Una alternativa de bajo costo y muy flexible es el uso de tarjetas plásticas de experimentación que consisten en una serie de conectores comunes en un medio aislante en el que se pueden realizar conexiones con mucha facilidad siendo éstas de carácter temporal, removibles y reutilizables (figura *2.19*).



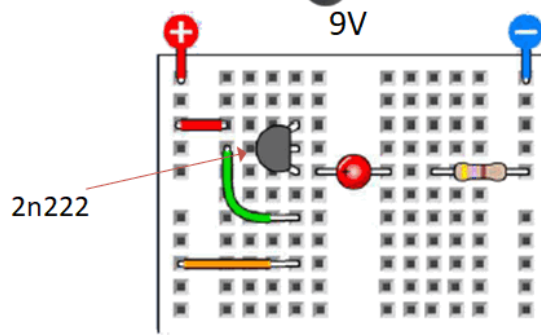


Figura 2.34 Placa de prueba y circuito ejemplo.

En la siguiente figura (figura *2.20*) resumimos algunos de los componentes eléctricos y electrónicos, así como sus diagramas. Note que no es de ninguna manera exhaustiva y estos símbolos son sólo una pequeña muestra de la gran cantidad que existe de ellos y que puede consultar en la bibliografía.

Nombre	Símbolo	Aspecto físico
Altavoz de imán permanente		
Antena exterior		
Auricular		
Capacitor		
Capacitor variable		
Capacitor electrolítico		
Diodo		
Diodo Emisor de Luz (LED)		
Diodo Zener		
Fonocaptor		
Fuente AC		
Fusible		
Inductor núcleo de aire		
Inductor núcleo de hierro		
Inductor variable Núcleo de hierro		
Interruptor	DPDT DPST	

Figura 2.35 Componentes eléctricos y electrónicos

Nombre	Símbolo	Aspecto físico
Lámpara Foco		
Micrófono		
Pila; batería		
Relevador		
Resistencia		
Resistencia Variable (Potenciómetro)		
Tierra		
Transformador núcleo de aire		
Transformador núcleo de hierro		
Transformador núcleo de hierro con devanado múltiple		
Transistor	Base Colector PNP Emisor Base Colector NPN Emisor	
Tubos de Vacío		

2.6 Resumen

Todo circuito eléctrico se forma de elementos o componentes que se analizan mostrando analogías hidráulicas sencillas para su comprensión. Sin estos elementos, la electrónica tal como la conocemos hoy, no sería posible.

El descubrimiento de materiales semiconductores es un paso importante hacia la integración en un área muy pequeña de todos los elementos eléctricos y a la formación de elementos nuevos llamados electrónicos que son capaces de ganancia (transistores).

2.6.1 Puntos Importantes del Capítulo

- La resistencia limita el paso de la corriente.

- Un capacitor almacena cargas en su campo eléctrico.
- Un inductor almacena carga en su campo magnético; esto se aprovecha para relevadores, filtros y transformadores, entre otros componentes.
- El relevador era usado en las primeras computadoras como elemento digital. Aún se usa, pero se prefiere su forma de estado sólido.
- El diodo solo permite el paso de la corriente en un sentido y es un dispositivo semiconductor.
- El diodo emisor de luz (***LED***) se usa como indicador preferentemente a su acción rectificadora.
- El transistor es un dispositivo semiconductor capaz de ganancia, pero que, en electrónica digital, se usa principalmente en su zona de saturación y corte como elemento digital.
- El modelo de Ebers-Moll da la relación entre corriente y ganancia de un transistor.
- Los circuitos integrados reúnen todos los componentes eléctricos y electrónicos en un solo elemento semiconductor, en un área sumamente reducida.
- Existen varias formas de interconectar los elementos de un circuito siendo la de uso más extendido la de los circuitos impresos.
- La tierra es indispensable para la interconexión de un circuito. La tierra física nos da un punto de referencia universal y único a partir del cual realizamos nuestras medidas, evitando tener accidentes.

2.7 Problemas

2.1 Observe cuidadosamente la figura *2.19*. Realice el diagrama eléctrico de la Placa de Prueba (breadboard) armado en la figura. Sabiendo que el componente 2n222 es un transistor común ¿Qué pasaría si conecta una pila de 9v alcalina tipo ***PP3*** (cuadrada) al circuito y usa su dedo humedecido con saliva para unir los cables naranja y verde? ¿Cambia el resultado si usa resistencias de distintos valores? Explique. ⚠ Atención: si desea hacer este experimento no exceda 9 voltios ni use otro tipo de pila que no sea alcalina. Si usa otro tipo de alimentación, no use su dedo sino una resistencia

2.2 Considere el circuito de la figura *2.3*. El ***LED*** emite luz cuando el voltaje V_{sal} es de 5V, en cuyo caso la diferencia de potencial a través del diodo emisor de luz es de 2 Voltios.

a) ¿Cuál es la corriente que fluye por el diodo emisor de luz si la resistencia es de 200Ω?

b) ¿Cuál es la corriente del ***LED*** si el voltaje cambia a 10V?

2.3 Explique por qué un capacitor que se pone entre las dos terminales de un contacto (switch) suprime la chispa que usualmente salta entre un interruptor al aproximarse lo suficiente y romper el dieléctrico del aire.

2.4 ¿Por qué un capacitor conectado entre la línea de potencia y tierra ayuda a eliminar las fluctuaciones de voltaje?

2.5 Investigue la forma de construcción de un transistor en silicio y haga un esquema.

2.6 Explique el funcionamiento del transistor en las regiones de corte y saturación.

2.7 ¿Por qué no se usa el transistor en su región lineal para propósitos de electrónica digital?

3 Sistemas Numéricos

Como matemático, Laplace apreciaba enormemente el sistema numérico decimal. Comprendió que cientos de años de esfuerzos mentales, así como de buena suerte, dieron como resultado el sistema que usamos. Laplace estaba en una posición en la que podía apreciar sus ventajas. Nuestro presente sistema numérico provee a los matemáticos modernos y científicos con grandes ventajas sobre los usados por anteriores civilizaciones y es un factor importante de nuestro rápido avance.



Pierre Simon Marqués de Laplace (1749-1827)

Astrónomo y matemático francés que dio base científica a la Hipótesis de las Nebulosas. Entre sus trabajos matemáticos más notables se destaca el perfeccionamiento de la teoría de las probabilidades.

Puesto que las manos son la herramienta más conveniente con la que la naturaleza nos dotó, los seres humanos siempre hemos tendido a usarlas para contar. Es entonces natural y afortunado que nuestro sistema de contar se base en el número de dedos que tenemos. Sin embargo, pasó bastante tiempo antes de poder representar estos números gráficamente. Las primeras representaciones gráficas encontradas consisten en marcas verticales y horizontales. El número 1 es un ejemplo de esto; es interesante hacer notar que el 2 consistía en dos marcas horizontales unidas con una línea y el 3 de tres marcas horizontales unidas (sistema arábigo). Los números Romanos son un buen ejemplo de líneas usadas como base para representar números[22] (ver figura *3.1*).

Decimal	0	1	2	3	4	5	6	7	8	9	10^6
Arábico-Índico	٠	١	٢	٣	٤	٥	٦	٧	٨	٩	
Bengali		১	২	৩	৪	৫	৬	৭	৮	৯	
Chino/Japonés	零	一	二	三	四	五	六	七	八	九	
Cuneiforme		𒁹	𒈫	𒐈	𒐉	𒐊	𒐋	𒐌	𒐍	𒐎	
Devanagari (Hindi)	०	१	२	३	४	५	६	७	८	९	
Egipcio		ı	ıı	ııı	ıııı	ııı ıı	ııı ııı	ıııı ııı	ıııı ıııı	ııı ııı ııı	𓁨
Griego		α	β	γ	δ	ε	ς	ξ	η	θ	
Hebreo		א	ב	ג	ד	ה	ו	ז	ח	ט	
Maya	𝋠	𝋡	𝋢	𝋣	𝋤	𝋥	𝋦	𝋧	𝋨	𝋩	
Romano		I	II	III	IV	V	VI	VII	VIII	IX	
Siamés		๑	๒	๓	๔	๕	๖	๗	๘	๙	
Tamil		௧	௨	௩	௪	௫	௬	௭	௮	௯	
Tibetano		༡	༢	༣	༤	༥	༦	༧	༨	༩	

Figura 3.36 Comparación de algunos sistemas numéricos.

El sistema decimal ha sido tan aceptado y adoptado por nuestra civilización que rara vez consideramos la posibilidad de otros sistemas en uso. De todas formas, no es razonable pensar que un

[22] Note que muchas civilizaciones desconocían el uso del cero.

sistema basado en el número de dedos que tenemos en las manos sea el más eficiente para usar en las máquinas que construimos. El hecho es que un sistema muy poco usado para otra cosa, pero muy sencillo, el sistema binario, ha probado ser el más natural y eficiente para su uso en máquinas computadoras.

Ya Charles Babbage[23] intentó durante 34 años de su vida fabricar un mecanismo basado en el sistema decimal; a partir de 1834 se obstinó en el diseño de su Máquina Analítica programable que debía tener 30m de largo por 10m de ancho. Su diseño se inspiraba en el telar de Joseph Marie Jacquard[24]. Se basada en 10 valores discretos usando engranes etiquetados del 0 al 9 muy similares al odómetro de un coche. Su intento fracasó por problemas técnicos de la época. Aún si lo hubiese logrado, imagine el problema de alinear y ajustar engranes para detectar su lugar en 10 posiciones ¿no hubiese sido más fácil y eficiente usando sólo dos?

3.1 El Sistema Decimal

Nuestro sistema actual se forma por 10 símbolos distintos: 0, 1, 2, 3, ..., 9 llamados Arábigos. Con este esquema nos veríamos forzados a detenernos en 9 o inventar otros símbolos para los demás números si no fuese porque usamos una notación de posición. Un ejemplo de esto lo podemos encontrar en los números romanos que son, en esencia, aditivos: III=I+I+I, XXV=X+X+V. Se requieren nuevos símbolos conforme la serie crece (X, L, C, D, M, etc.), así, se usa V en lugar de IIIII=5. La única importancia en la posición de los números Romanos es si un símbolo precede o antecede a otro (VI=6, IV=4). La torpeza de este sistema resalta en cuanto deseamos realizar cualquier operación con dos números, por ejemplo, multiplicar XII por XIV (12x14). Ni siquiera hablemos de fracciones. El cálculo con números Romanos es tan engorroso que los primeros matemáticos se vieron forzados a usar casi exclusivamente el ábaco o tablas de contar y luego traducir el resultado a números Romanos. Los cálculos con papel y lápiz son tan increíblemente complicados en este sistema que la habilidad para hacerlo era muy apreciada entre los antiguos Romanos.

La gran simplicidad y belleza de nuestro sistema decimal puede entonces apreciarse en detalle. Sólo es necesario aprenderse 10 dígitos y el sistema de notación de posición para contar cualquier cantidad. Después de aprender de memoria 10 tablas de

[23] Ver la introducción.
[24] Ver la introducción

multiplicar y las de sumar y unas sencillas reglas, es posible realizar cualquier operación aritmética. Nótese la sencillez para realizar la operación 12 por 14 en sistema decimal:

$$\begin{array}{r} 14 \\ \underline{x\ 12} \\ 28 \\ 14\ \ \\ \hline 168 \end{array}$$

El significado del número 168 puede notarse al decir la cantidad "ciento sesenta y ocho". Básicamente el número consiste en $(1x10^2)+(6x10^1)+(8x10^0)$. La importancia es que el valor de cada dígito está determinado por su posición. El 3 en 3000 vale distinto que el 3 en 30 y se denota al hablar diciendo tres mil o treinta (tres decenas).

La regla general para representar cualquier número en notación decimal es:

(3.1)

$$a_1 10^{n-1} + \cdots + a_2 10^{n-2} + a_n = a_1 a_2 \cdots a_n$$

donde n denota el número de dígitos a la izquierda del punto decimal.

La base del sistema se define como la cantidad de dígitos distintos usados en cada posición en un sistema de notación. El sistema decimal tiene base 10, esto significa que el sistema tiene 10 dígitos distintos para representar cualquier cifra (0, 1, 2, 3, ..., 9). La historia registra el uso de varias bases. El sistema quinario (base 5) prevalece entre los esquimales y los indios de Norte América; la base doce aún se usa en relojes, pies, docenas; el sistema base 60 (usado por los Babilonios) en segundos y minutos.

3.2 El Sistema Binario



Gottfired Wilhelm Baron von Leibnitz (1646-1716)

Filósofo y matemático alemán erudito en ciencia, historia y derecho. Desarrolló el cálculo infinitesimal sin conocer la obra de Newton en el mismo campo. Su filosofía se apoya fundamentalmente en la concepción de un universo compuesto por un número infinito de unidades de fuerza espiritual o materia a la que llama mónadas.

El matemático del siglo XVII, Leibnitz, era un fanático del uso de la base 2 que sólo usa los símbolos 0 y 1 para representar cifras. Puede parecer extraño que un matemático eminente use un sistema tan sencillo, pero debe recordarse que en esa época casi todos los matemáticos eran también filósofos y religiosos. Su preferencia al sistema base dos se debió a razones míticas, el uno representando a la deidad y el cero a la nada.

Cualesquiera que fuesen las razones de Leibnitz para usar el sistema binario, en los últimos años se ha vuelto muy popular. Todas las computadoras modernas se construyen para operar usando el sistema binario o sistemas codificados en binario y todo indica que en un futuro cercano seguirán siendo construidas de esta forma.

Los componentes básicos de las primeras computadoras eran los relevadores y contactos que son binarios por naturaleza pues sólo pueden estar en dos estados posibles: cerrados (1) o abiertos (0). Los principales componentes de las computadoras actuales son los transistores similares a los usados en televisores y radios. La necesidad de un funcionamiento confiable llevó a los diseñadores a utilizar a los transistores en sus estados de corte y saturación, reduciendo así sus estados posibles a dos fácilmente identificables, conduciendo (1) o no (0). Una simple analogía puede realizarse entre estos dos estados y un foco de luz eléctrica. En cierto momento el foco está prendido (transistor conduciendo) emitiendo luz o apagado (transistor no conduciendo). Aunque el foco esté viejo y no produzca tanta luz, se puede decir con certeza si está prendido o apagado. Lo mismo sucede con un radio, si éste está viejo o con las pilas gastadas, hay que compensar subiendo el volumen, pero por muy bajo que esté, siempre se puede decir si está prendido o no.

Debido al gran número de partes que forman una computadora, es altamente deseable utilizarlas de tal forma que los cambios en sus características no afecten el desempeño total del sistema. La mejor forma de lograr esto es usando los circuitos en su forma biestable (de dos estados posibles).

3.2.1 Contando en el Sistema Binario

El mismo tipo de notación de posición que usamos en el sistema decimal es el que se usa en el sistema binario. Obsérvense los primeros 16 números en el sistema binario:

Tabla 3.1			
Conversión de Decimal a Binario			
Decimal	Binario	**Decimal**	Binario
0	0000	**8**	1000
1	0001	**9**	1001
2	0010	**10**	1010
3	0011	**11**	1011
4	0100	**12**	1100
5	0101	**13**	1101
6	0110	**14**	1110
7	0111	**15**	1111

En la tabla anterior representamos los números binarios usando cuatro dígitos base 2 (2^4=16). Es importante que se acostumbre a hacerlo de esa forma lo más rápido posible por las razones que pronto comprenderá en el diseño de circuitos digitales y en la arquitectura de las computadoras.

Debe notarse que el sistema decimal usa potencias de 10, mientras que el binario, potencias de 2. En general el sistema ***n*** usará potencia ***n***. Mientras que el número 35 en decimal equivale a 3 x 10^1 + 5 x 10^0, el mismo número en binario (35) se representa como 100011_2 que significa 1 x 2^5 + 0 x 2^4 + 0 x 2^3 + 0 x 2^2 + 1 x 2^1 + 1x2^0. Los siguientes ejemplos ilustran la conversión de una base a la otra; debe notarse que la potencia puede substituirse por potencia ***n*** para sistema ***n*** (por ejemplo 16 para sistema base hexadecimal u ocho para sistema octal).

111 = 1 x 2^2 + 1 x 2^1 + 1 x 2^0 = 4 + 2 + 1 = 7
1011 = 1 x 2^3 + 0 x 2^2 + 1 x 2^1 + 1 x 2^0 = 6 + 0 + 2 + 1 = 9
11.011 =1 x 2^1+1 x 2^0+0 x 2^{-1}+1 x 2^{-2}+1 x 2^{-3} = 2 + 1 + 0 + ¼ + 1/8 = 3 3/8

Los números fraccionarios se forman al igual que en el sistema decimal:

0.135 = 1 x 10^{-1} + 3 x 10^{-2} + 5 x 10^{-3}

3.2.2 Conversión de Sistema Decimal a Binario

Existen muchos métodos de conversión entre bases, pero el primero y más obvio es restar todas las potencias de la base al número en base 10 hasta que el resultado sea cero. En base 2 restaremos la potencia mayor de 2 del número; al restante se le aplica el mismo procedimiento hasta que el resultado sea cero. Para convertir 25_{10} a base 2 restaremos de 25 la potencia mayor de 2 que no exceda a 25, en este caso 2^4 quedando 25-16=9 del que restamos la siguiente potencia mayor (2^3=8) y así continuamos hasta que no tengamos nada que restar. El número queda como $2^4 + 2^3 + 2^0$ o 0011 0001. Note el uso de 8 dígitos binarios para la representación del número, aunque los dos primeros ceros no se requieren. Acostúmbrese a hacerlo así, en grupos de 4. La razón será obvia según avance en la lectura del libro.

Un método más empleado para números grandes es dividir entre la base y apuntar el residuo resultante de derecha a izquierda, volviendo a dividir el resultado entre la base hasta que el resultado sea cero. Por ejemplo 125_{10}:

125÷2=62+1, 62÷2=31+0, 31÷2=15+1, 15÷2=7+1, 7÷2=3+1, 3÷2=1+1, 1÷2=0+1

tomando los residuos de derecha a izquierda: $0111\ 1101_2$.

En el caso de fracciones, se debe dividir el número en dos partes; la entera, en la que se aplica cualquiera de los métodos antes expuestos, y la fraccionaria.

La conversión de fracciones a la base de interés se logra restando de ésta las potencias negativas de la base hasta que no tengamos residuo o hasta la precisión que se requiera. Este primer método es engorroso para fracciones grandes por lo que se prefiere multiplicar por la base y tomar lo que queda a la izquierda del punto para formar la fracción y lo que queda a la derecha para volver a aplicar el procedimiento. Por ejemplo 0.4375_{10} a base 2:

2x0.4375=0.8750, 2x0.8750=1.750, 2x0.750=1.50, 2x0.50=1.0

tomando los números a la izquierda del punto decimal de izquierda a derecha tenemos que $0.4375_{10}=0.0111_2$.

3.2.3 Suma y Resta en Sistema Binario

La suma y resta en sistema binario son mucho más sencillas de aprender que en cualquier otro sistema, pues las reglas son muy sencillas. Para la suma tenemos que

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 0 y se lleva 1

Unos ejemplos bastan para comprender el procedimiento que es similar al decimal, al que ya estamos acostumbrados:

$$\begin{array}{r} 101 \\ +110 \\ \hline 1011 \end{array} \qquad \begin{array}{r} 1111 \\ +10100 \\ \hline 0010\,0011 \end{array} \qquad \begin{array}{r} 11.11 \\ +101.11 \\ \hline 1001.10 \end{array}$$

La resta tiene también reglas muy sencillas:

0 − 0 = 0
1 − 0 = 1
1 − 1 = 0
0 − 1 = 1 con un préstamo de 1

Y la forma de hacerla es similar al sistema decimal:

$$\begin{array}{r} 1001 \\ -101 \\ \hline 0100 \end{array} \qquad \begin{array}{r} 10000 \\ -11 \\ \hline 1101 \end{array} \qquad \begin{array}{r} 101.01 \\ -100.01 \\ \hline 0001.00 \end{array}$$

3.2.4 Multiplicación y División Binaria

La tabla de multiplicar usada por el sistema binario sólo tiene cuatro reglas a diferencia de las 100 usadas para la multiplicación en sistema decimal:

0 × 0 = 0
1 × 0 = 0
0 × 1 = 0
1 × 1 = 1

Sólo es necesario copiar de nuevo el multiplicando si se multiplica por 1 o poner ceros si es por 0:

$$\begin{array}{r} 1100 \\ \times\, 1010 \\ \hline 0000 \\ 1100 \\ 0000 \\ 1100 \\ \hline 0111\,1000 \end{array} \qquad \begin{array}{r} 0110\,0110 \\ \times\, 1000 \\ \hline 0011\,0011\,0000 \end{array}$$

Nuevamente la división es sumamente sencilla:

$$0 \div 1 = 0$$
$$1 \div 1 = 1$$

A continuación, dos ejemplos de la división:

```
          101                 10.011010101...
101 /11001        1100/11101.00
       101               1100
          101              10100
          101                1100
                               10000
                                 1100
                                   10000
                                     1100...
```

3.3 Representando Números en Otras Bases

Hemos dicho ya que los números en otras bases tienen tantos símbolos como la base de que se trate, esto es, en base dos tenemos dos símbolos distintos; en base 10, diez símbolos distintos para representar cualquier número. ¿Pero qué símbolos usar para bases mayores que diez? Distintas culturas han empleado distintos símbolos para resolver este problema, pero actualmente la única base usada, mayor que la decimal, es la base 16 o ***hexadecimal***.

Para representar números en base 16 usamos los nueve símbolos ya conocidos 0, 1, 2, ..., 9 y agregamos las letras para representar al 10, 11, 12, 13, 14 y 15. Así, la ***A*** representa al símbolo 10, ***B*** al símbolo 11, etc.

Para convertir de base 10 a cualquier otra empleamos los mismos métodos de la base dos, es decir dividiendo consecutivamente entre la base y anotando el residuo. Para el caso de base 16 dividiremos entre dieciséis.

Encontramos la equivalencia de un número en cualquier base con respecto a la decimal usando notación de posición y multiplicando por la base elevada a la potencia de la posición menos la unidad:

$F095_{16} = (15 \times 16^3) + (0 \times 16^2) + (9 \times 16^1) + (5 \times 16^0) = 61589_{10}$

Como la base 16 es divisible entre la base dos, el sistema hexadecimal es una forma conveniente de representar en notación corta al sistema binario evitando errores (con este mismo propósito se utilizó mucho tiempo la base 8). La forma de convertir de sistema hexadecimal a binario es sustituir cada uno de los símbolos usados en el número en hexadecimal por su equivalencia en binario:

$$F095_{16} = 1111\ 0000\ 1001\ 0101_2$$

o cualquier número en binario dividiendo el número en grupos de 4 de derecha a izquierda y sustituyendo por su símbolo equivalente en base 16:

$$10001010110_2 = 0100\ 0101\ 0110_2 = 456_{16}$$

Para estas conversiones es conveniente tener una tabla a la mano:

Tabla 3.2			
Tres Distintas Representaciones de los Enteros del 0 al 15.			
Decimal	***Binario***	***Hexadecimal***	***BCD***
00	0000	0	0000 0000
01	0001	1	0000 0001
02	0010	2	0000 0010
03	0011	3	0000 0011
04	0100	4	0000 0100
05	0101	5	0000 0101
06	0110	6	0000 0110
07	0111	7	0000 0111
08	1000	8	0000 1000
09	1001	9	0000 1001
10	1010	A	0001 0000
11	1011	B	0001 0001
12	1100	C	0001 0010
13	1101	D	0001 0011
14	1110	E	0001 0100
15	1111	F	0001 0101

3.4 Decimal Codificado en Binario (BCD)

Puesto que las computadoras construidas que usan el sistema binario requieren de una menor cantidad de circuitos electrónicos y por lo tanto son más eficientes que las máquinas que operan con otros sistemas numéricos, el sistema binario es el sistema más natural para una computadora y el de mayor uso actualmente. Por otro lado, el sistema decimal es el más natural para nosotros. Todos los cálculos que realizamos usualmente se realizan en el sistema decimal, pero deben ser convertidos por las computadoras de decimal a binario antes de realizar cualquier operación. Debido a esto, muchas de las primeras computadoras usaban un sistema de codificación decimal a binario. En tal sistema, se usan grupos de dígitos binarios para representar cada uno de los 10 símbolos usados en el sistema decimal. Por ejemplo, uno de los códigos más obvios y naturales es usar un ***código binario de pesos*** donde cada posición representa un ***peso*** tal y como se muestra en la tabla *3.3.*

Tabla 3.3

Código binario de pesos.

Código Binario	Dígito Decimal
Peso	
8 4 2 1	
0 0 0 0	0
0 0 0 1	1
0 0 1 0	2
0 0 1 1	3
0 1 0 0	4
0 1 0 1	5
0 1 1 0	6
0 1 1 1	7
1 0 0 0	8
1 0 0 1	9

Nótese que son necesarios 4 dígitos binarios para cada símbolo decimal. Esto es ineficiente pues las combinaciones de 4 dígitos binarios son $2^4 = 16$ de los que sólo usamos 10 pero si usamos 3 dígitos $2^3 = 8$ son insuficientes.

Para simplificar la conversión de decimal a binario, es más práctico codificar un número decimal $d_{n-1}\, d_{n-2} \ldots\, d_1\, d_0$ donde $d_i \in \{0, 1, \ldots, 9\}$ de la siguiente forma: Substituya cada dígito decimal con su equivalente de 4 símbolos binarios tal como están definidos en la tabla

2 anterior. El número resultante es llamado código binario decimal, código 8, 4, 2, 1 o simplemente ***BCD*** (Binary-coded decimal), y lo indicaremos con el sufijo ***BCD***. Así, tenemos que:

$$0100\ 0101\ 0101_{BCD} = 455_{10}$$

Nótese que el número resultante ***BCD*** y el binario son bastante distintos. A cada uno de los dígitos binarios le llamamos ***bit*** (de binary digit). En la representación ***BCD***, cada segmento de 4 bits representa un sólo dígito decimal cuyo peso es una potencia de 10. Únicamente 10 de los 16 posibles patrones de grupos de 4 bits son necesarios para la representación ***BCD***. Aunque el formato ***BCD*** claramente simplifica la conversión decimal, tiene la desventaja de requerir más bits por cada número a representar. Usando 8 bits, el mayor número representable es

$$1001\ 1001_{BCD} = 99_{10}$$

mientras que en binario con el mismo número de bits tenemos:

$$1111\ 1111_{2} = 255_{10}$$

Otras de las desventajas que presenta este sistema es en las operaciones matemáticas como la resta que normalmente se realizan usando el complemento del número (ver siguiente sección). Para resolver estas desventajas se diseñaron otros tipos de códigos. Uno de los primeros es el llamado de ***exceso 3*** (excess 3) en el que, para formar la equivalencia, primero se suma 3 al número decimal. Por ejemplo, para representar al 4, primero sumamos 3, resultando 7 y luego usamos el ***BCD*** "normal", que es 0111. El 0111 es el código exceso 3 para el 4.

Cambiando cada 0 por 1 y cada 1 por 0 formamos lo que se llama el complemento del número binario. Este procedimiento es usado para formar el complemento a 9 de un número decimal. Por ejemplo, el complemento de 0100 (1 decimal en código exceso 3) es 1011 u ocho decimal.



Howard Hathaway Aiken (1900-1973)

Físico nacido en EE.UU. pionero en la computación. Sus diseños llevaron a la construcción de la 1era calculadora moderna: la Mark I de IBM (1944). Contaba con 3 millones de conexiones eléctricas y 800 km de cable, sólo hacia las 4 operaciones aritméticas básicas y funcionaba con tarjetas perforadas.

El código exceso 3 no es un código de peso (de notación), esto es, cada 1 no representa una potencia de 2 que podamos sumar para formar el número decimal. Un código de peso en el que el complemento a 9 pueda ser formado complementando cada bit, es el código 2, 4, 2, 1 o código Aiken representado en la tabla 4. Este código se usa extensivamente en relojes, instrumentos digitales y calculadoras electrónicas.

Tabla 3.4

Representaciones Alternas

Decimal	Exceso 3	Complemento a nueve	Código 2, 4, 2, 1
			Peso
			2 4 2 1
00	0011	1100	0 0 0 0
01	0100	1011	0 0 0 1
02	0101	1010	0 0 1 0
03	0110	1001	0 0 1 1
04	0111	1000	0 1 0 0
05	1000	0111	1 0 1 1
06	1001	0110	1 1 0 0
07	1010	0101	1 1 0 1
08	1011	0100	1 1 1 0
09	1100	0011	1 1 1 1

3.5 Números Negativos

Hasta el momento sólo hemos trabajado con números positivos (sin signo) pero el signo positivo o negativo es necesario para distinguir a los positivos de los negativos. Los números sin signo se consideran como positivos y el signo + es omitido. En una computadora los números se almacenan en una memoria que tiene un número finito y fijo de posiciones[25]. Cada posición puede tomar un valor de 0 o 1 y es costumbre representar a los números negativos reservando la última posición de la izquierda para el signo. De esta forma, una computadora que tenga longitudes de registro, llamadas palabras de 8 bits (8 bits equivalen generalmente a 1 byte) sólo podrán usarse 7 posiciones (128 números distintos o 2^n-1) y la última para el signo.

Por convención se ha escogido utilizar un 1 para números negativos y 0 para los positivos. De tal forma, -104 se representa por

[25] De 4, 8, 16, 32 o más bits.

$1110\ 1000_2$ y 104 como $0110\ 1000_2$. A esta notación se le llama notación con signo. Las operaciones matemáticas pueden realizarse en la misma forma que las operaciones manuales con números decimales.

El resultado de una operación matemática en una palabra de n bits requiere, típicamente, que el resultado sea una palabra de n bits. Si el resultado completo es una palabra de n+1 bits, como en el caso de una suma, se dice que ha ocurrido un ***desbordamiento*** o ***saturación*** (overflow). Un desborde se indica con una generación extra de una señal que modifica una memoria de un bit cambiándola de 0 a 1. En algunos casos la saturación puede ser ignorada y considerar como resultado adecuado a los n bits de la palabra y en otros será necesario tomar alguna acción correctiva.

Aunque el código de notación con signo es la contraparte directa del código decimal usado por los seres humanos, otra representación binaria llamada de ***complementos*** es usada con frecuencia en computadoras digitales, principalmente porque simplifica la construcción de ciertas operaciones aritméticas.

Existen dos tipos de complementos usados:

1. ***Complemento a la base***. Se forma restando cada dígito de la cantidad al número de la base menos uno y luego agregando 1 al resultado así obtenido. Para el sistema decimal llamamos a esta forma complemento a diez; para el binario, complemento a dos.
2. ***Complemento a la base menos 1***. Se forma restando a cada dígito de la cantidad al número de la base menos uno. Para el sistema decimal llamamos a esta forma complemento a nueve; para el binario, complemento a uno.

Por ejemplo, el complemento a diez de 87_{10} es 13_{10} (12_{10}+1_{10}) y el de 23_{10} es 77_{10} (76_{10}+1_{10}) mientras que el complemento a nueves de 87_{10} es 12_{10} y el de 77_{10} es 22_{10}. En el sistema binario el complemento a dos de $0001\ 0110_2$ es $0000\ 1010_2$ y el de $0001\ 1010_2$ es $0000\ 0110_2$.

El método para encontrar el complemento en la base dos de cualquier cantidad consiste en sustituir todo 0 por 1 y todo 1 por 0, dependiendo del tipo de complemento se sumará 1 o no.

La ventaja principal de usar complementos en sistemas digitales durante la suma o resta, es que todos los bits del número son tratados de forma uniforme y que la resta y suma son realizadas por el mismo circuito que sólo suma, simplificando, así, el diseño de la electrónica.

Al usar la representación por complementos la resta queda simplificada realizándose con sumas:

Complemento a la base

```
     89           89
    -23          +77
  -----        -----
     66           66  (el último acarreo se descarta)
  11011        11011
 -10100       +01100
  -----      -------
  00011      1 00111  (el último acarreo se descarta)
```

Complemento a la base menos uno

```
     89           89
    -23          +76
  -----        -----
     66         1 65  (el último acarreo se suma)
  11001           +1
 -10100        -----
  -----           66
  00011        11011
              +01001
             -------
             1 00010
                  +1  (el último acarreo se suma)
               -----
               00011
```

3.6 Código Gray y ASCII

Las secuencias de los números binarios son "naturales" y generalmente se comprenden con facilidad pues siguen un patrón posicional tal como el sistema decimal. Podemos, sin embargo, representar un número por una secuencia arbitraria de 1 y 0. Para evitar ambigüedad debemos, sin embargo, asignar a cada valor numérico una secuencia distintiva e individual.

Los números representados en otros sistemas que los naturales, son llamados ***códigos*** puesto que se les debe asignar una regla de asignación para determinar el valor numérico representado por la secuencia. Ya hemos analizado algunos códigos utilizados en computación e introducimos dos más: el Gray reflejado y el ***ASCII***.

El código ***ASCII*** (American Standard Code for Information Interchange, Código Americano Estándar para el Intercambio de Información) es un esfuerzo de los diseñadores para tener compatibilidad entre las distintas máquinas y aplicaciones. Este código se desarrolló en el ámbito de la telegrafía y se forma de 7 bits dejando el octavo disponible para que el diseñador juegue con él, ya sea para comprobar por medio de paridad (explicada en los siguientes capítulos) si no hay error de transmisión o manipulación de datos o para expandir el juego de caracteres disponibles elevándolo de 2^7 (128) a 2^8 (256) (esto último es lo que se realiza en las computadoras modernas). Las primeras 32 posiciones fueron pensadas y reservadas para caracteres de control de dispositivos que usaban ***ASCII*** y usualmente no se pueden desplegar o imprimir (hay formas de dar la vuelta a esto). Por ejemplo, el carácter 10 representa la función "nueva línea" (Line Feed o ***LF***), que hace que una impresora avance el papel, y el carácter 27 representa la tecla "escape" que a menudo se encuentra en la esquina superior izquierda de los teclados que los occidentales consideramos comunes[26].

Los demás se usan para las letras, los números y toda una serie de símbolos utilizados. En Japón y otros países donde se usa otro tipo de letras que no son las romanas, se utilizan 2 o más bytes (16 bits o posiciones de unos y ceros) para que el código pueda representar "todas" sus palabras o símbolos[27]. El usar este código garantiza, en muchos casos, compatibilidad entre datos de distintas aplicaciones de forma que el resultado de un programa pueda ser leído por otro con poco esfuerzo de nuestra parte.

---

[26] Cada país tiene una disposición distinta de teclado. Por ejemplo EE.UU. usa el llamado QWERTY llamado así por la disposición de la 1era fila de letras comenzando a partir de la esquina superior izquierda. En Francia se usa el AZERTY mientras que en Alemania el QWERTZ.

[27] El chino mandarín usa 56,000 símbolos lo que requiere una palabra de registro de 2^{16}. Una persona culta reconoce alrededor de 10,000. En lugar de usar ***ASCII*** se prefiere el UTF-16 o UTF-32 (Unicode de 32 bits).

Caracteres de control		
00	NULL	(Null character)
01	SOH	(Start of Header)
02	STX	(Start of Text)
03	ETX	(End of Text)
04	EOT	(End of Trans.)
05	ENQ	(Enquiry)
06	ACK	(Acknowledgement)
07	BEL	(Bell)
08	BS	(Backspace)
09	HT	(Horizontal Tab)
10	LF	(Line feed)
11	VT	(Vertical Tab)
12	FF	(Form feed)
13	CR	(Carriage return)
14	SO	(Shift Out)
15	SI	(Shift In)
16	DLE	(Data link escape)
17	DC1	(Device control 1)
18	DC2	(Device control 2)
19	DC3	(Device control 3)
20	DC4	(Device control 4)
21	NAK	(Negative acknowl.)
22	SYN	(Synchronous idle)
23	ETB	(End of trans. block)
24	CAN	(Cancel)
25	EM	(End of medium)
26	SUB	(Substitute)
27	ESC	(Escape)
28	FS	(File separator)
29	GS	(Group separator)
30	RS	(Record separator)
31	US	(Unit separator)
127	DEL	(Delete)

Caracteres Imprimibles					
32	space	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[	123	{
60	<	92	\	124	|
61	=	93	]	125	}
62	>	94	^	126	~
63	?	95	_		

Caracteres extendidos							
128	Ç	160	á	192	└	224	Ó
129	ü	161	í	193	┴	225	ß
130	é	162	ó	194	┬	226	Ô
131	â	163	ú	195	├	227	Ò
132	ä	164	ñ	196	─	228	õ
133	à	165	Ñ	197	┼	229	Õ
134	å	166	ª	198	ã	230	µ
135	ç	167	º	199	Ã	231	þ
136	ê	168	¿	200	╚	232	Þ
137	ë	169	®	201	╔	233	Ú
138	è	170	¬	202	╩	234	Û
139	ï	171	½	203	╦	235	Ù
140	î	172	¼	204	╠	236	ý
141	ì	173	¡	205	═	237	Ý
142	Ä	174	«	206	╬	238	¯
143	Å	175	»	207	¤	239	´
144	É	176	░	208	ð	240	≡
145	æ	177	▒	209	Ð	241	±
146	Æ	178	▓	210	Ê	242	‗
147	ô	179	│	211	Ë	243	¾
148	ö	180	┤	212	È	244	¶
149	ò	181	Á	213	ı	245	§
150	û	182	Â	214	Í	246	÷
151	ù	183	À	215	Î	247	¸
152	ÿ	184	©	216	Ï	248	°
153	Ö	185	╣	217	┘	249	¨
154	Ü	186	║	218	┌	250	·
155	ø	187	╗	219	█	251	¹
156	£	188	╝	220	▄	252	³
157	Ø	189	¢	221	¦	253	²
158	×	190	¥	222	Ì	254	■
159	ƒ	191	┐	223	▀	255	nbsp

Figura 3.37 Tabla ASCII de 8 bits

En el ***código Gray*** o ***código binario reflejado***, los dos primeros números son representados en forma natural; la siguiente serie de números son encontrados de la forma especificada en la figura *3.3*. Una imagen "espejo" se representa por la línea de guiones entre los primeros dos números (en la tabla de la izquierda) y que da lugar al punto de la reflexión. De ahí en adelante un cero es añadido a la primera serie de números y un 1 a la segunda. Repeticiones sucesivas del proceso de reflexión nos permiten hacer cualquier serie de números.

La característica más importante del código Gray es que los números cambian de uno a otro sólo en un dígito. Por ejemplo, notemos que del número 7 (0100) al 8 (1100) sólo cambia el dígito en la posición cuarta mientras que en la representación binaria (7=0111, 8=1000) cambian 4 dígitos. Esta característica encuentra aplicación en un sinnúmero de situaciones de la que daremos sólo un breve ejemplo[28].



Frank Gray
(1887-1969)

Físico e investigador nacido en EE.UU. que trabajó en los Laboratorios Bell dónde realizó numerosas innovaciones en el área de la televisión. Inventor del código que lleva su nombre. Junto con otros, proveyó el código de la TV por ***PCM*** (Pulse Code Modulation).

Supongamos que tenemos una veleta que nos dará la dirección del viento en cualquier momento. Un circuito recibe la señal y la interpreta mandando el resultado a un computador que analizará los datos junto con muchos más para dar un pronóstico de tiempo. Si el circuito detecta que más de un dígito cambia a la vez, podemos estar seguros de que hay un error en la transmisión o en el sensor y podemos descartar la lectura esperando a la siguiente o dar la alarma para que la situación se corrija.

Primeros
dos números

00
01
———— Reflejo
11
10

0 00
0 01
0 11
0 10
———— Reflejo
1 10
1 01
1 00

Decimal	Binario	Gray
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

Reflejo

Figura 3.38 Código Gray.

3.7 Resumen

Los números son la base de todas las representaciones formales que realizamos. Las bases distintas a la de diez han estado en uso durante mucho tiempo, pero es la base dos la única distinta de 10 que actualmente usamos en circuitos digitales. Dada su sencillez, la base dos se utiliza en las computadoras digitales actuales. Es

[28] Entre otras aplicaciones podemos citar: corrección de errores, rompecabezas matemáticos, comunicación entre relojes digitales, contadores y aritmética digital.

importante entenderla y ser capaz de realizar fácilmente operaciones sencillas.

3.7.1 Puntos Importantes del Capítulo

- El sistema decimal es un sistema de posiciones en la que a cada sitio se le da un peso que equivale a potencias de 10.
- El sistema binario se usa extensivamente en computación por su sencillez y equivalencia a los dos estados de un transistor (corte y saturación) y diodo.
- Existen otras representaciones de números usando unos y ceros que hacen más fácil el tratamiento de números dentro de la computadora como son el ***BCD***, exceso tres, 2421 y complementos.
- Los números negativos se representan reservando un bit para el signo.
- El código Gray, ***ASCII*** y UTC son usados extensivamente en computación y en muchas otras aplicaciones.

3.8 Problemas

3.1 Convierta los siguientes números a su equivalente binario:

a) 39_{10}, b) 12_{10}, c) 123_{10}

3.2 De los números binarios obtenidos en el problema 3.1 encuentre su equivalente hexadecimal.

3.3 Convierta los números del problema ***3.1*** a base 8 y base 4 (recuerde que la base 8 sólo consta de 8 símbolos distintos de 0 a 7, y la base 4 del 0 al 3).

3.4 Encuentre la equivalencia binaria de los siguientes números:

a) 3.123_{10}, b) 0.4375_{10}, c) 1.1_{10}

3.5 Sume y luego reste en forma binaria 12_{10} + 10_{10}.

3.6 Divida y multiplique en forma binaria 3_{10} y15_{10}.

3.7 Convierta los siguientes números binarios a su complemento de 1 y de 2:

a) $0001\ 0111_2$, b) 1001_2, c) 1111_2

3.8 Haga la siguiente resta usando complementos a 1 y a 2: $0001\ 1101_2 - 0001\ 1011_2$

3.9 Convierta los siguientes números hexadecimales a binario:

a) BCD_{16}, b) 635_{16}, c) FFF_{16}

3.10 Una regla sencilla para multiplicar dos números en cualquier base es multiplicar los dos números en forma decimal. Si el producto es menos que la base, se toma tal cual; si es mayor que la base, divida en decimal entre la base y tome el residuo como el dígito menos significativo y el cociente como el más significativo. Por ejemplo $2_6 \times 2_6 = 4_6$, $3_8 \times 2_8 = 6_8$; pero $2_6 \times 4_6 = 8_6$ y $6 \div 8 = 1$ y sobran 2 por lo que $2_6 \times 4_6 = 12_6$. Usando esta regla, multiplique:

a) $2_7 \times 4_7$, b) $2_7 \times 3_7$, c) $5_4 \times 4_4$

4 Circuitos Lógicos

4.1 Introducción

En los sistemas digitales encontramos variables que son especiales en el sentido de que sólo pueden tomar dos valores posibles. Por ejemplo, en sistemas electrónicos digitales, un voltaje o corriente típico tiene una forma de onda (idealmente) muy similar a la de la figura *4.1*.

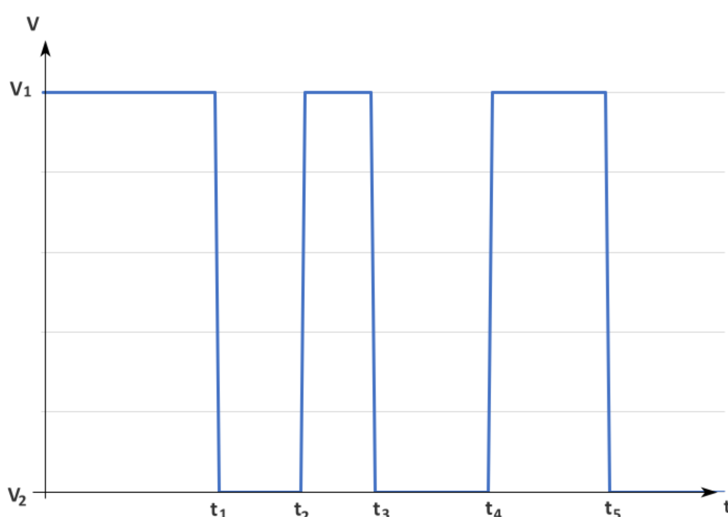


Figura 4.39 Forma de onda idealizada de un sistema digital.

En esta figura observamos que el voltaje ***V*** tiene cambios abruptos entre dos niveles de voltaje V_1 y V_2. Idealmente consideramos estas transiciones, que suceden en tiempos $t = t_1, t_2$, etc., tan abruptas que podemos decir que en todo momento $V = V_1$ o $V = V_2$ y que ***V*** no tiene ningún otro valor intermedio. De forma alternativa podemos decir que, si las transiciones de ***V*** no son tan abruptas, los valores que ***V*** pueda tener no nos interesan más que cuando $V = V_1$ o $V = V_2$. En la figura *4.1* indicamos que V_1 es positivo mientras que V_2 es negativo. Esta característica no es esencial y puede ser que tanto V_1 como V_2 sean positivos o negativos; o puede suceder que ya sea V_1 o V_2 sean cero.

Podemos especificar el valor de la función *V(t)* en cualquier instante diciendo que $V = V_1$ o $V = V_2$; o, por la misma característica especial de ***V***, decir que ***V*** está "alto" o "bajo", "arriba" o "abajo", etc. Más aún, después de haber seleccionado dos palabras,

asignamos arbitrariamente una palabra a un nivel de voltaje y la otra al segundo nivel. Para sugerir que un voltaje puede tener sólo un valor a la vez en un tiempo ***t***, es importante seleccionar las palabras de forma que sean mutuamente exclusivas y opuestas. Por estas razones y otras más presentadas a continuación, escogeremos las palabras "verdadero" (***V***) y "falso" (***F***) para designar los dos estados. Es necesario especificar si verdadero especifica el caso en que $V = V_1$ o $V = V_2$. Claro está, que falso designa el otro estado posible. Si arbitrariamente decidimos que el voltaje más positivo es verdadero y el más negativo es falso, hemos adoptado una lógica que llamaremos ***lógica positiva***. Si adoptamos la convención contraria estaremos hablando de ***lógica negativa***.

En un sistema digital particular encontraremos muchas variables binarias (de dos valores) tal como la variable ***V*** de la figura *4.1*. Debemos entender que, en ese sistema dado, todas las variables binarias operan entre las mismas alternativas. Por ejemplo, si en un sistema eléctrico las variables binarias son los voltajes V_a y V_b, entonces cada voltaje operará entre los mismos dos niveles de voltajes, digamos +5 Volts y 0 Voltios (ejemplo de valores reales empleados en circuitos digitales).

4.2 Funciones de una sola Variable Binaria

Sea A una variable binaria y ***Z*** una segunda variable binaria que es una función de A, entonces: ***Z=f(A).***

¿Cuáles son las posibles relaciones funciónales entre ***Z*** y ***A***? Puesto que ***A*** únicamente puede tomar el valor de verdadero o falso (***V*** o ***F***) y ***Z*** es también ***V*** o ***F***, sólo existen dos posibles funciones. Una posibilidad es cuando ***A*** es verdadera, ***Z*** sea verdadera y que cuando ***A*** no es verdadera (***A*** falsa) ***Z*** sea también falsa. En este caso escribiremos: ***Z=A***.

El otro caso ***Z*** puede asumir la alternativa de ***A***, esto es, si ***A*** es verdadera, ***Z*** es falsa y si ***A*** es falsa, ***Z*** es verdadera, por lo que: $Z = \bar{A}$ y se lee como ***Z*** es igual a 'no ***A'*** o ***Z*** es igual al 'complemento de ***A'***. Las dos ecuaciones anteriores agotan todas las posibilidades de la función ***Z*** de una sola variable binaria.

Debe notarse que el complemento del complemento de una variable es la variable misma:

$$\bar{\bar{A}} = A \qquad (4.1)$$

4.3 Funciones de dos Variables Binarias

Consideremos el caso en que la variable binaria ***Z*** depende de dos variables binarias ***A*** y ***B***, esto es, ***Z=f(A,B)***. Una función se define dando las reglas o cualquier otra información por la que la variable dependiente ***Z*** puede ser determinada cuando las variables independientes ***A*** y ***B*** son especificadas. En el caso de variables continuas debemos dar la ecuación que las defina, pero para el caso binario sólo existen cuatro posibilidades que son las variaciones con repetición de dos variables tomadas de dos en dos en dónde sí importa el orden y sí se repiten los elementos, esto es $VR_m^n = m^n = 2^2 = 4$. Es por lo tanto posible definir la variable ***Z*** especificando los valores de ***Z*** para todas las variaciones de ***A*** y ***B***. Definiremos a continuación dos distintas funciones ***f(A,B)*** y ***g(A,B)*** especificadas en forma tabular en la siguiente tabla (*4.1*):

Tabla 4.1			
Dos funciones de las variables A y B			
A	**B**	**Z=f(A, B)**	**Z=g(A, B)**
F	F	F	F
F	V	F	V
V	F	F	V
V	V	V	V

En esta tabla se listan las variaciones posibles de ***A*** y ***B***. A estas tablas y otras similares se les llama tablas de verdad y las funciones dadas en la tabla anterior fueron escogidas entre otras porque son de especial interés e importancia (ver siguientes secciones).

4.3.1 Operación Y (AND)

La función ***f(A, B)*** de la tabla *4.1* se puede expresar en palabras como sigue: ***Z*** es verdadero si ***A*** es verdadera ***y B*** es verdadera. Por lo que a esta función se le llama ***Y*** (*AND* en inglés) y la relación funcional se escribe ***Z = A y B***. La operación ***Y*** también se denota con una notación más conveniente con un punto (por razones explicadas más adelante) para dar la idea de multiplicación, aunque no se intente ninguna multiplicación numérica. Frecuentemente el punto se omite y la operación ***Y*** se escribe como:

$$(4.2) \qquad Z = A \text{ y } B = A \cdot B = A \wedge B = A \cap B = A\,B$$

Se puede verificar en la tabla *4.1* que la operación es conmutativa:

$$(4.3) \qquad AB = BA$$

La operación ***Y*** es también asociativa por lo que si tenemos el resultado ***Z=AB*** y una tercera variable ***C*** interviene, tenemos que:

$$(4.4) \qquad (AB)C = A(BC) = (AC)B$$

Por lo que, si muchas variables intervienen en la operación ***Y***, no necesitamos indicar el orden de las operaciones y podemos escribir simplemente ***Z=ABCD...***

El símbolo usado para representar la conexión de un número de variables por medio de una operación ***Y*** se muestra en la figura *4.2a*. Tal dispositivo, que realiza la conexión entre variables, se llama ***compuerta lógica***. Es de especial interés para nosotros el caso de que una variable representa voltajes. En tal caso el intento de la figura *4.2a* es indicar que si cada una de las terminales ***A, B, C, D*** se mantienen a un voltaje seleccionado como verdadero (con respecto a una tierra que no es indicada explícitamente), entonces la salida ***Z*** también es verdadera; si medimos el voltaje veremos que se encuentra al mismo que se seleccionó como 1 o verdadero. Si cualquiera de las entradas está en el voltaje opuesto (escogido como falso), la salida será también falsa. Nótese que la compuerta ***Y*** tiene una situación de simultaneidad y coincidencia, esto es, todos los voltajes de entrada deben ser simultánea o coincidentemente verdaderos para que la salida sea verdadera. De acuerdo con esto, la compuerta ***Y*** se conoce también como de ***coincidencia***.

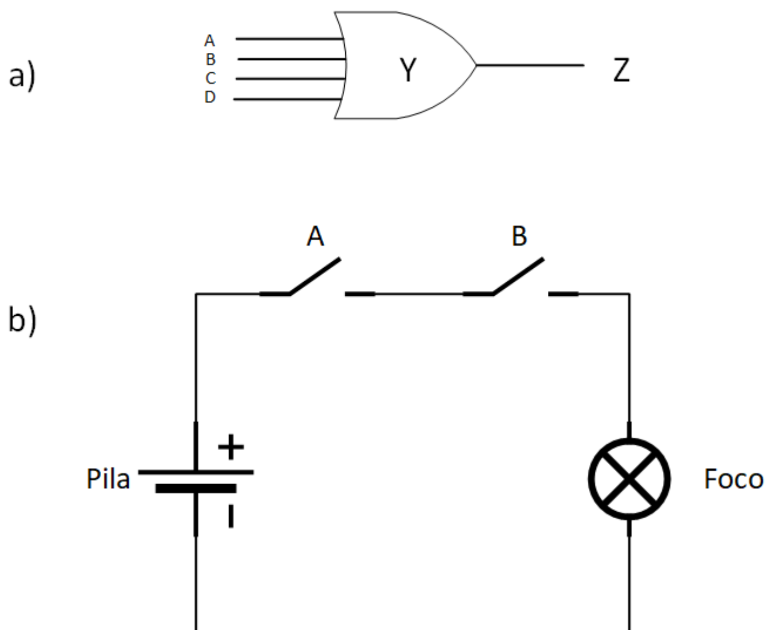


Figura 4.40 Compuerta tipo Y (AND).

4.3.1.1 La Operación *Y* con Interruptores

Hemos resaltado que un voltaje que tiene sólo dos posibles estados se puede representar con una variable lógica. Un interruptor puede ser representado con una variable lógica. El interruptor tiene dos estados posibles: cerrado o abierto. Podemos asignar al interruptor una variable lógica ***A*** entendiendo que si ***A=F*** (falso) significa que el interruptor está abierto y que si ***A=V*** (verdadero) el interruptor está cerrado. O, si así nos conviene, podemos asignar los estados contrarios (***A=F*** cerrado y ***A=V*** abierto).

La operación ***Y*** realizada por un circuito eléctrico con interruptores se muestra en la figura *4.2b*. Sea ***A=V*** si el interruptor S_1 está cerrada, ***B=V*** si el interruptor S_2 está cerrado y ***Z=V*** si la luz se prende. Puesto que la luz sólo se prende si S_1 y S_2 se cierran, la operación del circuito se describe con la ecuación lógica ***Z=AB***. Si arbitrariamente hacemos la asignación de que ***A=F*** si S_1 está cerrado, *Z* se convierte entonces en $Z = \bar{A}B$.

4.3.2 Operación O (OR)

La función ***g(A,B)*** de la tabla *4.1* define la operación ***O*** (*OR* en inglés). Lo apropiado de la palabra ***O*** puede verse en el hecho de que la función ***Z*** sólo es verdadera cuando ***A*** es verdadera o ***B*** es verdadera o si *A* y ***B*** son verdaderas. La función se escribe como:

(4.5) $$Z = A \; o \; B = A \vee B = A \cup B = A + B$$

El signo + representa la operación ***O*** y de ninguna forma implica una suma algebraica. Pero su uso (así como el de la multiplicación para la operación ***Y***) junto con otra notación que a continuación describimos, es nemotécnico.

Se puede verificar que la operación ***O***, tal como la ***Y***, es conmutativa y asociativa por lo que:

(4.6) $$A + B = B + A; (A + B) + C = A + (B + C)$$

El símbolo usado para la operación se indica en la figura *4.3a*. La operación puede lograrse con interruptores tal como se indica en la figura *4.3b*. La luz se prende si cualquiera de los interruptores se cierra.

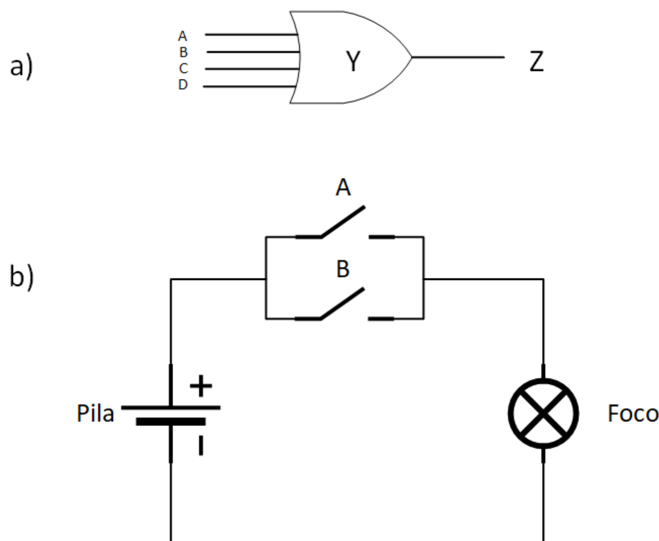


Figura 4.41 Compuertas tipo NO Y (NAND) y NO O (NOR).

4.3.3 Operación NO Y y NO O (NAND y NOR)

La operación ***NO Y*** (***NAND*** en inglés) se representa simbólicamente con la ecuación:

(4.7) $$Z = A \uparrow B \quad o \quad \overline{AB}$$

y es simplemente el complemento o negación de la función ***Y*** ya definida anteriormente. Si observamos la siguiente tabla (*4.2*):

Tabla 4.2						
Funciones de dos variables						
A	**B**	Y	O	NO Y	NO O	O EXCLUSIVA
F	**F**	F	F	V	V	F
F	**V**	F	V	V	F	V
V	**F**	F	V	V	F	V
V	**V**	V	V	F	F	F

Se debe notar que (de la columna 5):

$$Z = \overline{AB} = \bar{A} + \bar{B} \quad (4.8)$$

Que es un ejemplo del ***Teorema De Morgan*** analizado en la sección *4.7*.

La operación ***NO Y*** es conmutativa, pero no asociativa, como puede verificarse de las tablas de verdad:

$$\overline{AB} = \overline{BA}; \quad (4.9)$$
$$(A \uparrow B) \uparrow C \neq A \uparrow (B \uparrow C) \; o$$
$$(\overline{\overline{AB}) C} \neq \overline{A(\overline{BC})}$$

Si la operación ***NO Y*** se aplica a más de una variable se usa en el sentido $Z = \overline{ABCD \ldots}$ que no es ambigua mientras que $Z = A \uparrow B \uparrow C \uparrow D \ldots$ no está definida hasta que se le agreguen paréntesis a la expresión.

El símbolo de la compuerta ***NO Y*** es igual al de una ***Y***, pero se agrega un pequeño círculo a la salida para indicar la negación u operación ***NO*** (***NOT*** en inglés; ver figura *4.4a*).



Augustus De Morgan
(1806-1871)

Matemático y lógico inglés cuyas principales contribuciones al estudio de la lógica incluyen la formulación del teorema que lleva su nombre y trabajos fundamentales que llevan al desarrollo de la teoría de las relaciones y el surgimiento de la lógica moderna o matemática simbólica.

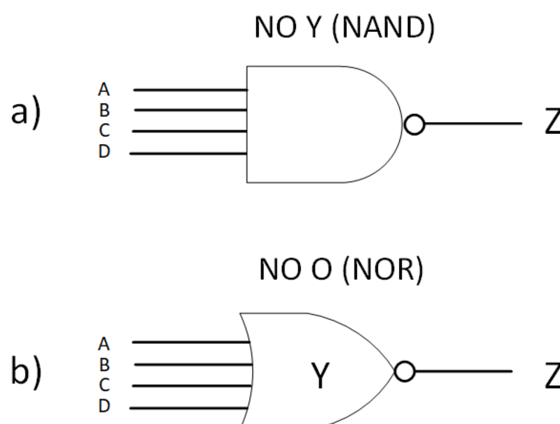


Figura 4.42 Compuerta tipo O (OR).

La compuerta ***NO O*** (***NOR*** en inglés) es la negación de la ***O*** y su función se representa por:

$$(4.10) \qquad Z = A \downarrow B \quad o \quad \overline{A+B}$$

Si observamos la tabla *4.2* en su sexta columna tenemos la tabla de verdad de una ***NO O***. La ***NO O*** se puede expresar también como:

$$(4.11) \qquad Z = \overline{A+B} = \bar{A}\bar{B}$$

otro caso del teorema de Morgan.

Tal como la ***NO Y***, la ***NO O*** es conmutativa pero no asociativa:

$$(4.12) \qquad \overline{A+B} = \overline{B+A};$$
$$(A \downarrow B) \downarrow C \neq A \downarrow (B \downarrow C) \quad o$$
$$\overline{\overline{A+B} + C} \neq \overline{A + \overline{B+C}}$$

Si la operación ***NO O*** se aplica a más de una variable se entiende en el sentido de $Z = \overline{AB} = \bar{A} + \bar{B}$.

El símbolo para la compuerta ***NO O*** es igual al de una ***O,*** pero se usa un pequeño círculo a la salida para indicar la negación (ver figura *4.4b*).

4.3.5 La operación O EXCLUSIVA

La operación ***O EXCLUSIVA*** (***XOR*** o *Exclusive Or* en inglés) que conecta a dos variables se representa simbólicamente por la ecuación:

$$(4.13) \qquad Z = A \oplus B = A\bar{B} + \bar{A}B$$

y su tabla de verdad se define en la tabla *4.2* última columna. Expresado en palabras, ***Z*** es verdadera si ***A*** o ***B*** son verdaderas en forma exclusiva, esto es, cuando ni ***A*** ni ***B*** son verdaderas simultáneamente.

Una forma de implementar la función de ***O EXCLUSIVA*** consiste en el circuito de la figura *4.5a* (vea el problema *4.2* para un circuito con interruptores).

Se puede verificar que la operación ***O EXCLUSIVA*** es conmutativa y asociativa y más aún, la función ***Z*** es verdadera cuando hay un

número impar de variables verdaderas y falsa si hay un número par de variables verdaderas.

La compuerta se representa como se indica en la figura *4.5a*. Resulta que agregar variables (señales de entrada) a una compuerta electrónica real ***O***, ***Y***, ***NO O*** o ***NO Y*** es muy sencillo como analizamos en el capítulo *6*, pero para una ***O EXCLUSIVA***, el problema no es tan sencillo y aunque en la figura *4.5b* se muestra el diagrama de una ***O EXCLUSIVA*** de 4 entradas, sólo existen compuertas comerciales con dos entradas. Para agregar entradas se usa la disposición esquematizada en la figura *4.5c*.

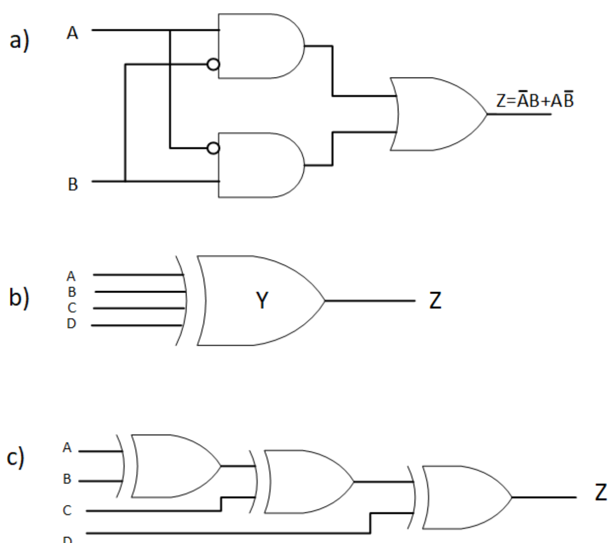


Figura 4.43 Compuerta tipo O Exclusiva (Exclusive OR).

4.3.6 Otras Funciones

La función ***Z=f(A,B)*** de dos variables consta de 16 posibles funciones de las que hemos considerado sólo 6 funciones: ***NO*** (negación o complemento), ***Y***, ***O***, ***NO O***, ***NO Y***, ***O EXCLUSIVA***. Las restantes 10 aunque de interés, es posible realizarlas con las que ya hemos analizado (ver más adelante en este capítulo).

4.4 Variables Lógicas

A las variables de dos valores que hemos presentado hasta el momento, se les conoce como variables lógicas, mientras que a las operaciones como la ***Y*** o la ***O*** se les conoce como operaciones lógicas. Analizaremos la importancia de tal terminología y al hacerlo remarcaremos la ventaja de designar a los estados posibles como verdadero y falso.

Algebra

Forma generalizada de la aritmética en la que se incluyen símbolos para representar parámetros o cantidades desconocidas logrando, así, una generalización.



Muḥammad ibn Mūsā al-Khwārizmī

(780-850)

Matemático, astrónomo y astrólogo persa cuyo nombre da origen a la palabra algoritmo y el título de una de sus obras, Kitâb al-jabr wa al-muqâbala, a algebra. Padre del algebra y primero en divulgar el uso del cero y el sistema decimal posicional, prestado de la India.



George Boole

(1815-1864)

Matemático Inglés que ayudo a establecer la lógica simbólica moderna y cuya álgebra de lógica, ahora llamada álgebra Booleana, es la base del diseño de circuitos de computación digital.

Pongamos el caso de un automóvil que se debe proteger por medio de una alarma. Si alguna de las puertas es abierta, la persona, basándose en la habilidad lógica de razonar, deduce que se está tratando de perpetrar un robo y que la alarma debe sonar. Si cada una de las puertas de acceso al automóvil se alambra por medio de un circuito que tenga un voltaje V_1 cuando el interruptor que controla la puerta esté cerrado y un voltaje V_2 al estar abierto, y le asignamos arbitrariamente verdadero al voltaje V_2, por lo que V_1 es falso, el circuito debe tener la propiedad de que su voltaje de salida sea V_2 cuando cualquiera de los accesos al coche sea abierto. Hemos construido un circuito del tipo ***O*** que deduce lógicamente que un acceso fue abierto y su salida V_2 activa una sirena para avisarnos.

Si las variables ***A, B, C, ...*** representan un acceso abierto, tenemos que ***Z*** es verdadera cuando cualquier puerta se abre: *Z=A+B+C+...*

Está claro que $\bar{A}, \bar{B}, \bar{C}$,etc. representan la proposición contraria.

Hemos escogido representar a las variables lógicas ***A***, ***B*** y ***Z*** como niveles de voltaje, pero se debe hacer notar que sólo representan relaciones entre proposiciones y son independientes de la forma en que queramos representarlas físicamente.

Esta algebra de proposiciones es conocida como ***Algebra Booleana*** y así como otras algebras tratan con variables que tienen significado numérico, el álgebra de Boole trata con proposiciones y es una herramienta efectiva para analizar relaciones entre proposiciones que permitan sólo dos estados mutuamente excluyentes.

Ejemplo

4.1 Un granjero llamado Juan tiene un perro, una cabra y hierba que tiene guardada en dos almacenes que llamaremos Norte y Sur. A Juan no le es posible dejar a la cabra sola con la hierba pues se la comería, así como tampoco se puede quedar el perro sólo con la cabra pues la mataría. Juan nos encarga diseñar una computadora portátil que ilumine una lámpara en caso de emergencia y le permita decidir qué debe llevar con él para evitar el desastre de dejar al perro con la cabra o a la cabra con la hierba. ¿Cómo construimos el circuito?

Respuesta

Para diseñar el circuito debemos establecer precisamente en qué casos se enciende la lámpara:

1. Juan está en el almacén norte ***Y*** el perro ***Y*** la cabra están en el almacén sur, ***O*** si
2. Juan está en el almacén norte ***Y*** la cabra ***Y*** la hierba en el sur, ***O*** si
3. Juan está en el almacén sur ***Y*** el perro ***Y*** la cabra están en el almacén norte, ***O*** si
4. Juan está en el almacén sur ***Y*** la cabra ***Y*** la hierba en el norte.

Consideremos que la variable ***J*** representa a Juan en el almacén sur, por lo que $\bar{J}$ representa Juan en el almacén norte. En este sentido tenemos también que:

P perro en almacén sur
$\bar{P}$ perro en almacén norte
C cabra en almacén sur
$\bar{C}$ cabra en almacén norte
H hierba en almacén sur
$\bar{H}$ hierba en almacén norte

Podemos escribir una función lógica que combine todas las posibilidades que nos lleven a un desastre:

$$Z = \bar{J} \cdot P \cdot C + \bar{J} \cdot C \cdot H + J \cdot \bar{P} \cdot \bar{C} + J \cdot \bar{C} \cdot \bar{H}$$

donde ***Z*** representa la luz que indica un posible desastre. La figura *4.6* nos da la solución en un circuito funcional.

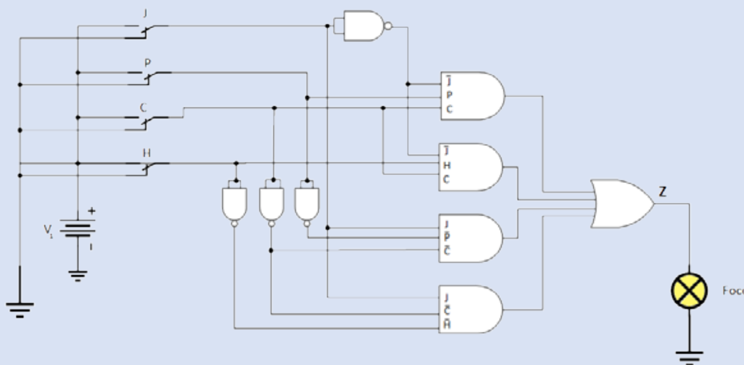


Figura 4.44 Computadora especial del ejemplo 4.1.

4.5 La Notación 0 y 1

Nos hemos referido frecuentemente al tipo de notación donde tenemos dos posibilidades mutuamente exclusivas que pueden tomar cualquier valor y lo hemos asignado arbitrariamente como verdadero o falso. Proponemos ahora alternativamente la notación 0 para falso y 1 para verdadero. Se hace notar que 0 y 1 son símbolos que representan falso y verdadero y no son usados en ningún momento como números.

Aún más, usaremos como ya habíamos propuesto en las secciones anteriores, el signo de + para indicar la operación ***O*** y el de multiplicación para la operación lógica ***Y***. Las tablas respectivas de ambas operaciones lógicas quedan como se muestra en la tabla *4.3*:

Tabla 4.3	
Función O e Y con notación alterna +, •	
O	**Y**
0+0=0	$0 \cdot 0 = 0$
0+1=1	$0 \cdot 1 = 0$
1+0=1	$1 \cdot 0 = 0$
1+1=1	$1 \cdot 1 = 1$

Notemos que la nueva notación es un efectivo dispositivo nemotécnico, puesto que, si pretendemos que la suma y la multiplicación se realizan de verdad, todas las reglas algebraicas se aplican con la excepción de *1+1=1*.

4.6 Operaciones Necesarias y Suficientes

De las 16 funciones posibles de 2 variables (2^4) hemos analizado específicamente 5 de ellas y se ha observado que todas (incluyendo las no analizadas) pueden expresarse en términos de operaciones ***Y***, ***O*** y *NO*.

Si llevamos el asunto hasta sus últimos extremos podemos ver que no es necesario usar las tres operaciones puesto que la ***Y***, se puede expresar en función de la *NO* y la ***O***, o la ***O*** es posible expresarla en términos de *NO* e ***Y*** (figura *4.7a* y *4.7b*).

Aunque en principio la operación *NO* junto con, ya sea la ***O*** o la ***Y***, son suficientes, en el diseño y análisis lógico de sistemas usaremos las tres pues es más conveniente de esta forma.

Por último, es de especial interés ver el uso de las compuertas ***NOY*** y ***NOO*** para generar operaciones *NO*, ***O*** e ***Y*** tal como se muestra en las figuras *4.7c*, *4.7d*, *4.7e*, *4.7f*, *4.7g* y *4.7h*.

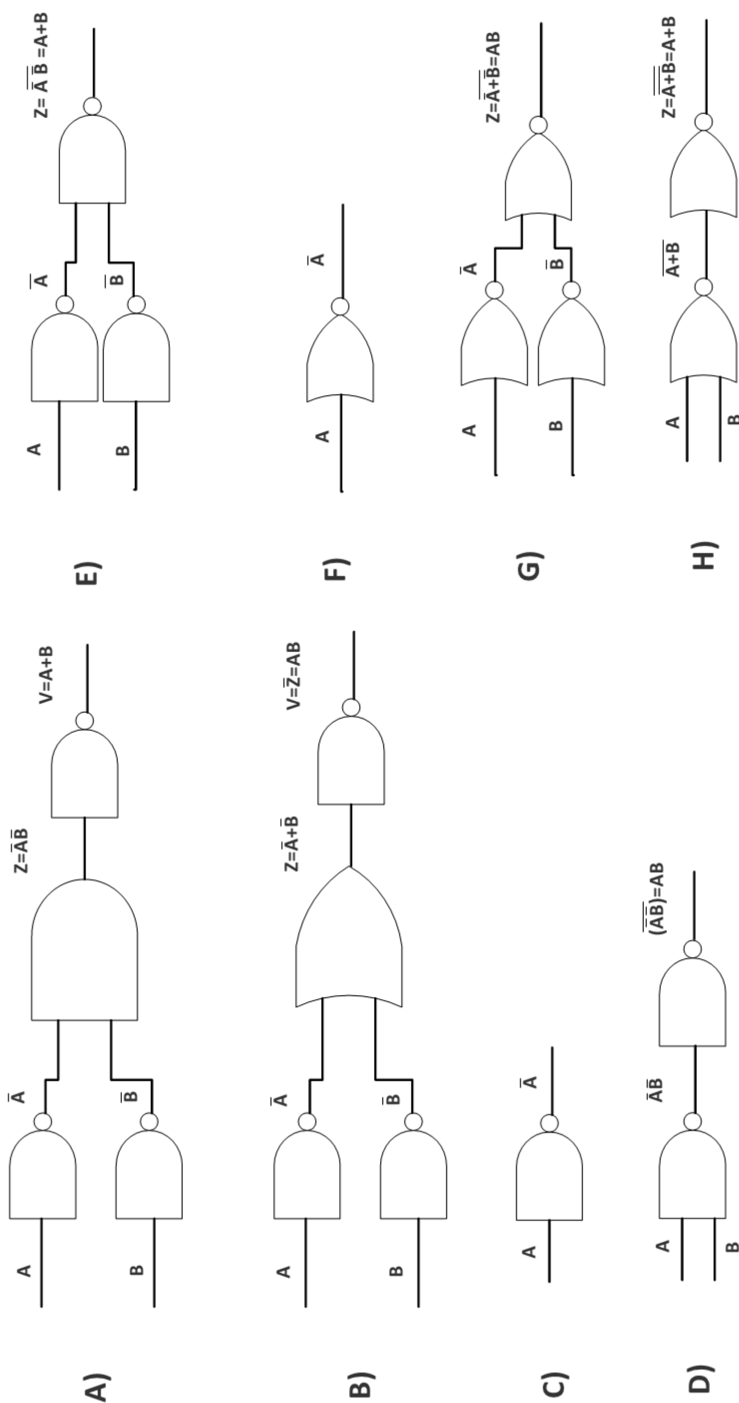


Figura 4.45 Operaciones necesarias y suficientes.

4.7 Teoremas del Algebra Booleana

Teorema

Proposición o aseveración que tiene que ser probada por razonamiento lógico de hechos comprobables.

Enumeraremos ahora una serie de teoremas que nos serán de gran utilidad para simplificar expresiones con variables lógicas. Restringiremos nuestra atención a los teoremas que tratan sobre las operaciones lógicas ***O***, *NO* e ***Y***. Usaremos el símbolo de multiplicación y suma para representar los conectivos, así como la notación ***0*** y ***1*** para denotar falso y verdadero.

Notemos inicialmente que los resultados de la negación son evidentes por el hecho de que una variable sólo puede tener dos posibles valores ***A=0*** o ***A=1***, por lo que (ya visto en *4.1*):

$$\bar{\bar{A}} = A \qquad (4.14)$$

puesto que si ***A=1 entonces*** $\bar{A} = 0$ y $\bar{\bar{A}} = 1$ nuevamente. De forma similar $\bar{0} = 1$ y $\bar{1} = 0$.

También tenemos que:

(4.15a)	$A + 0 = A$	(4.15b)	$A \cdot 1 = A$
(4.16a)	$A + 1 = 1$	(4.16b)	$A \cdot 0 = 0$
(4.17a)	$A + A = A$	(4.17b)	$A \cdot A = A$
(4.18a)	$A + \bar{A} = 1$	(4.18b)	$A \cdot \bar{A} = 0$

Estos primeros ocho teoremas son sólo de una variable. En la prueba de los teoremas se puede tomar ventaja del hecho que ***A*** sólo puede tener dos valores posibles, por lo que podemos probar analizando todas las posibilidades y comprobando que el teorema es válido en todos los casos. Por ejemplo, para el primer caso tenemos que si *A=1, 1+0=1*, que es correcto. Si *A=0, 0+0=0* que una vez más es correcto.

Hemos colocado los teoremas en dos columnas para resaltar la característica de que el de la derecha es el ***dual*** del de la izquierda (o viceversa). Dado un teorema, su dual se encuentra:

1. Intercambiando el signo + por •.
2. Intercambiando los ceros por unos.

Esta dualidad no debe ser sorpresa pues si observamos las tablas de las funciones ***O*** e ***Y*** vemos que una es el dual de la otra.

La ley distributiva se aplica al álgebra de variables lógicas de forma que *A(B+C) = AB+AC*. En su forma dual la ley distributiva aparece como:

(4.19)	$A+BC=(A+B)\,(A+C)$
(4.20)	$A(B+C) = AB+AC$

Nótese una vez más que el lado izquierdo es el dual del derecho. La primera ecuación (*4.19*) parece errónea a primera vista, pero se deja como ejercicio al lector comprobar su validez por el método exhaustivo.

Enunciamos ahora otros teoremas usuales de dos variables con sus duales:

(4.21a)	$A + AB = A$	(4.21b)	$A(A + B) = A$
(4.22a)	$A + \bar{A}B = A + B$	(4.22b)	$A(\bar{A} + B) = AB$
(4.23a)	$AB + A\bar{B} = A$	(4.23b)	$(A + B)(A + \bar{B}) = A$
(4.24a)	$AB + \bar{A}C = (A + C)(\bar{A} + B)$	(4.24b)	$(A + B)(\bar{A} + C) = AC + \bar{A}B$
(4.25a)	$AB + \bar{A}C + BA = AB + \bar{A}C$	(4.25b)	$(A + B)(\bar{A} + C)(B + C) = (A + B)(\bar{A} + C)$

Finalmente, enunciamos un teorema adicional llamado ***Teorema de Morgan*** en sus dos formas (una el dual de la otra):

(4.26a)	$\overline{A \cdot B \cdot C \cdots} = \bar{A} + \bar{B} + \bar{C} + \cdots$
(4.26b)	$\overline{A + B + C + \cdots} = \bar{A} \cdot \bar{B} \cdot \bar{C} \cdots$

Que si lo expresamos en palabras dicen:

(1) El complemento del producto de variables es igual a la suma del complemento de cada una de las variables.
(2) El complemento de la suma de variables es igual al producto del complemento de cada una de las variables.

Ejercicios

4.1 Pruebe que el teorema *4.26a* es correcto (use las tablas de verdad de ***A***, ***B*** y ***C*** y luego, combinándolas, demuestre el teorema).

4.2 Por el mismo método empleado en el ejercicio anterior demuestre el teorema *4.26b*.

Ejemplo

4.2 Suponga que un estudiante universitario quiere inscribirse al curso de Arquitectura de Computadoras y encuentra que sólo puede inscribirse si cumple con los siguientes requisitos:

1. Ha completado por lo menos 60 créditos y es un estudiante de la carrera MAC (Matemáticas Aplicadas a La Computación) con buen promedio o
2. Ha completado 60 créditos, es estudiante de MAC y lo aprueba el departamento o
3. Tiene menos de 60 créditos y es un estudiante de MAC que no tiene buen promedio o
4. Tiene buen promedio y la aprobación del departamento o
5. Es un estudiante de MAC, aunque no tenga la aprobación del departamento.

Respuesta

Veamos como toda esta serie de restricciones pueden simplificarse usando el álgebra Booleana.

Introduzcamos las variables:

- ***A*** Estudiante con por lo menos 60 créditos.
- ***B*** Estudiante de MAC.
- ***C*** Estudiante con buen promedio.
- ***D*** Estudiante con aprobación del departamento.
- ***Z*** función que, si se cumple, indica que puede tomar el curso de Arquitectura de Computadoras.

Las variables sólo pueden tomar el valor de cierto o falso (1 o 0), esto es, si *C=1* indica que el estudiante tiene buen promedio y si *C=0* no tiene buen promedio. Podemos establecer todas las condiciones del punto 1 al 5 en la siguiente ecuación:

$$Z = ABC + ABD + \bar{A}B\bar{C} + CD + B\bar{D}$$

Combinando el término segundo con el quinto y aplicando la ley distributiva (teorema *4.20*):

$$Z = ABC + \bar{A}B\bar{C} + CD + B(\bar{D} + DA)$$

Del teorema *4.22a* tenemos que $\bar{D} + DA = \bar{D} + A$:

$$Z = ABC + \bar{A}B\bar{C} + CD + B\overline{D} + AB$$
$$= ABC + AB + \bar{A}B\bar{C} + CD + B\overline{D}$$

Factorizando AB de los primeros dos términos y notando que *C+1=1* (teorema *4.16a*):

$$Z = AB(C + 1) + \bar{A}B\bar{C} + CD + B\overline{D}$$
$$= AB + \bar{A}B\bar{C} + CD + B\overline{D}$$

Factorizando ***B*** de los primeros dos términos vemos que $A + \bar{A}\bar{C} = A + \bar{C}$ (del teorema 4.22a):

$$Z = B(A + \bar{A}\bar{C}) + CD + B\overline{D} = AB + B\bar{C} + CD + B\overline{D}$$

Notemos del teorema *4.25a* que dada una expresión $CD + B\overline{D}$ que aparece en por lo menos dos términos, podemos agregar el término BC sin cambiar el valor de la expresión. Agregando tal término y recombinando con el segundo de la última ecuación tenemos que:

$$Z = AB + B\bar{C} + BC + CD + B\overline{D}$$
$$= AB + B(\bar{C} + C) + CD + B\overline{D}$$

o

$$Z = AB + B + CD + B\overline{D}$$

y combinando los términos primero, segundo y cuarto:

$$Z = B(1 + A + \overline{D}) + CD$$

Sabemos que *X+1=X* y si tomamos a ***X*** como $A + \overline{D}$ finalmente llegamos a:

$$Z = B + CD$$

Esto es, se puede tomar el curso si es un estudiante de MAC ***O*** si tiene buen promedio ***Y*** si tiene permiso del departamento.

Notará en el ejemplo *4.2* que el método empleado para simplificar la ecuación original deja mucho que desear y se debe tener un amplio dominio de todos los teoremas, así como gran ingenio y habilidad para realizar agrupaciones que lleven a una (posible) simplificación. Es por eso que se han diseñado muchos otros

métodos alternativos para realizar dichas simplificaciones. Estudiaremos algunos de esos métodos en el siguiente capítulo.

4.8 Resumen

El capítulo plantea los principios fundamentales del algebra de Boole estableciendo los principios que lo regulan y los teoremas que nos auxilian a un diseño digital posterior. Se introduce el concepto de variable lógica y varios circuitos bases de los que construiremos todos los componentes de una computadora a lo largo del libro.

4.8.1 Puntos Importantes del Capítulo

- Las variables lógicas sólo pueden tomar dos valores mutuamente excluyentes.
- Con las funciones ***O***, ***Y***, *NO* se puede representar todas las demás.
- La notación empleando 0 y 1 así como los signos de + y · simplifican la notación y son nemotécnicos.
- Existen operaciones necesarias y suficientes.
- Con los teoremas del algebra Booleana podemos simplificar ecuaciones que plantean problemas complejos.

4.9 Problemas

4.1 Dibuje un circuito que permita encender una luz si los interruptores *A* o *B* se cierran o ambos se cierran.

4.2 Con un circuito similar al de la figura *4.3b* realice un circuito para las funciones lógicas ***NOY*** y ***NOO***.

4.3 Muestre cómo el circuito de la figura *4.8* puede representar una función ***O EXCLUSIVA***. Los interruptores deben estar unidos para que sólo tengamos dos variables independientes.

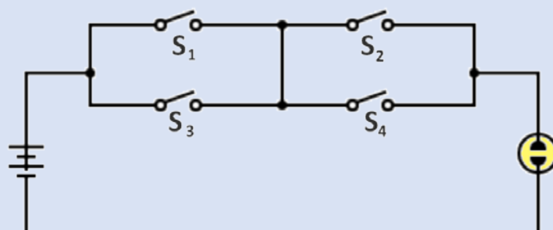


Figura 4.46 Sencillo circuito que simula una compuerta XOR.

4.4 Emiliano planea ir al cine si Cristina va con él y si puede usar el coche de la familia. Sin embargo, Cristina planea ir al club si no llueve y si la temperatura es mayor a 20ºC. El padre de Emiliano planea usar el coche para visitar a sus amigos si llueve o si la temperatura es mayor a 20ºC. ¿Bajo qué condiciones irá Emiliano al cine? Construya una computadora especial con circuitos ***Y***, ***O*** y *NO*, interruptores, batería y una luz que se encienda si Emiliano va al cine.

4.5 Encuentre el dual de las siguientes ecuaciones:

a) $\bar{A}B$

b) $(\bar{A}+B)(\bar{C}\bar{D})$

c) $(A+\bar{A}B)(C+\bar{D})$

4.6 Simplifique las siguientes ecuaciones:

a) $(A+B)\bar{A}\bar{B}\bar{C}$

b) $A\bar{B}C+\bar{A}\bar{C}D+\bar{C}A$

c) $AB+\bar{A}C\bar{D}E+\bar{B}C\bar{D}$

d) $\bar{A}\bar{B}+AC+\bar{B}\bar{C}\bar{D}+B\bar{C}E+\bar{B}CF+BC\bar{G}$

5 Simplificación de Funciones Lógicas

En el capítulo anterior analizamos las funciones de una y dos variables lógicas, así como las operaciones lógicas que con ellas podemos realizar. Vimos también los teoremas del álgebra de Boole y sus implicaciones en la reducción de expresiones lógicas complejas; se llegó a la conclusión de que era necesario desarrollar un método más eficaz de hacer estas reducciones de una forma sistemática y sencilla.

En este capítulo analizaremos algunas de estas formas.

5.1 Formas Estándares de las Funciones Lógicas

En vistas de encontrar un procedimiento para desarrollar formas de simplificar funciones lógicas, introducimos en este punto dos formas estándares en las que las funciones lógicas pueden ser expresadas.

5.1.1. La Suma Estándar de Productos

La ***suma estándar de productos*** es aquella en la que las variables lógicas que intervienen en una función son sumadas en la función, multiplicándose entre ellas.

Ejemplo

5.1 Dada la función lógica de cuatro variables

$$f(A,B,C,D) = (\bar{A} + BC)(B + CD)$$

exprese la función como una suma de productos.

Respuesta

Usando la ley distributiva tenemos que:

$$f(A,B,C,D) = (\bar{A} + BC)B + (\bar{A} + BC)CD$$

$$= \bar{A}B + BBC + \bar{A}CD + BCCD$$

$$= \bar{A}B + BC + \bar{A}CD + BCD$$

5.2 Dada la función lógica de cinco variables

$$f(A,B,C,D,E) = (A + \overline{BC})(\overline{D + BE})$$

exprese ***f*** como una suma de productos.

Respuesta
Usando el teorema De Morgan y la ley distributiva tenemos que:

$$f(A,B,C,D,E) = (A + \bar{B} + \bar{C})[\bar{D}(\overline{BE})]$$

$$= (A + \bar{B} + \bar{C})[\bar{D}(\bar{B} + \bar{E})]$$

$$= (A + \bar{B} + \bar{C})(\bar{B}\bar{D} + \bar{D}\bar{E})$$

$$= A\bar{B}\bar{D} + A\bar{D}\bar{E} + \bar{B}\bar{D} + \bar{B}\bar{D}\bar{E} + \bar{B}\bar{C}\bar{D} + \bar{C}\bar{D}\bar{E}$$

En los ejemplos anteriores hemos visto cómo una expresión lógica arbitraria puede ser escrita como la suma de productos. Si sólo las variables individuales aparecen complementadas como en el primer ejemplo, necesitamos aplicar únicamente la ley distributiva. Si se complementa una combinación de variables, como en el segundo ejemplo, debemos aplicar primero el teorema De Morgan.

De todas formas, siempre es posible escribir una expresión lógica como una simple suma de términos, siendo cada término el producto de alguna combinación de variables, algunas complementadas y otras no. La misma variable nunca debe aparecer dos veces en el mismo producto, puesto que sabemos que una repetición de una variable o del complemento de una variable puede ser eliminada usando los teoremas (*4.17b*, *4.1*, *4.18b*) $AA = A$, $\bar{A}\bar{A} = \bar{A}$ o $A\bar{A}$=0.

Podemos notar que en las expresiones de la suma de los productos no necesariamente intervienen todas las variables en cada uno de los términos individuales (ver ejemplos anteriores). Una estandarización más que nos lleva a expresiones en que en todos los términos aparecen todas las variables (complementadas o no) se puede lograr siguiendo el procedimiento ilustrado en el siguiente ejemplo (*5.3*).

Ejemplo

5.3 Dada la función lógica de tres variables

$$f(A,B,C) = A + \bar{B}C$$

en que cada uno de los términos individuales no contiene a las tres variables, transforme de forma tal que cada uno de los términos de la función ***f*** contenga las tres variables.

Respuesta

Nótese que en el primer término no aparece ni la ***B*** ni la ***C***, pero podemos multiplicar por $(B + \bar{B})(C + \bar{C})$ que no cambia el significado pues sabemos que $B + \bar{B} = 1$. De forma similar multiplicamos el segundo término por $(A + \bar{A})$:

$$f(A,B,C) = A(B + \bar{B})(C + \bar{C}) + (A + \bar{A})(\bar{B}C)$$

$$= ABC + AB\bar{C} + A\bar{B}C + A\bar{B}\bar{C} + A\bar{B}C + \bar{A}\bar{B}C$$

de donde eliminamos al término duplicado $A\bar{B}C$:

$$f(A,B,C) = ABC + AB\bar{C} + A\bar{B}C + A\bar{B}\bar{C} + \bar{A}\bar{B}C$$

Las expresiones en las cuales la suma de los productos contiene a todas las variables, son llamadas ***suma estándar de los productos*** y a cada uno de los términos se les conoce como ***minitérminos***. El mérito de la suma estándar de productos es que cierta información acerca de las funciones lógicas está disponible inmediatamente después de una primera inspección. Puesto que la forma contiene la suma lógica de términos, la función tiene el valor de verdadero (***f = 1***) cuando uno (o más) de los términos tiene el valor lógico de ***1***. Considere el primer término de la ecuación resultante del ejercicio *5.3*. Este 1er término, ***ABC,*** siendo un producto, tendrá el valor lógico de ***1*** sólo cuando ***A=1, B=1*** y ***C=1***. De forma similar el segundo término ($AB\bar{C}$) nos dice que ***f=1*** cuando ***A=1, B=1***, $\bar{C} = 1$ o en forma equivalente cuando ***A=1, B=1*** y ***C=0.*** Cada minitérmino específica, entonces, una combinación de valores lógicos de cada una de las variables individuales por las cuales la función tiene el valor lógico de ***1***. Todos los minitérminos colectivamente especifican todas las combinaciones de valores de las variables para las cuales la función tiene el valor lógico de ***1.*** Con ***N*** variables tenemos 2^N minitérminos y si todos los minitérminos están presentes entonces ***f=1***.

Cada minitérmino corresponde a un renglón de la tabla de verdad de la función en el que la función ***f*** tiene el valor de 1; si por ejemplo examinamos una tabla de verdad de una función ***f*** de tres variables y notamos que en un renglón ***f=1*** cuando ***A=1, B=0*** y ***C=0***, sabremos que uno de los minitérminos será $A\bar{B}\bar{C}$. El número de minitérminos de la expresión ***f*** es igual al número de renglones de la tabla de verdad en donde la función ***f=1***.

5.1.2 El Producto Estándar de las Sumas

La segunda forma a analizar es conocida como el ***producto estándar de las sumas*** de una función y se expresa como un producto de términos en los cuales cada término consiste en la suma de todas las variables de la función (en forma complementada o no) multiplicada por otros términos.

En la sección anterior llegamos a la suma de productos estándar usando la regla $A + \bar{A} = 1$ y la ley distributiva. Podemos establecer una forma alternativa usando el dual de estas dos reglas. Si usamos la ley distributiva en la forma ***A+BC=(A+B) (A+C)*** y la regla $A\bar{A} = 0$ llegaremos al producto de las sumas.

Ejemplo

5.4 Dada la función lógica de tres variables

$$f(A,B,C) = A + \bar{B}C$$

exprese ***f*** como un producto de sumas.

Respuesta

Usando la ley distributiva tenemos que:

$$f(A,B,C) = (A + \bar{B})(A + C)$$

Si se quiere expresar como un producto estándar de la suma tenemos que agregar a cada uno de los términos las variables faltantes. Recuerde que $X\bar{X} = 0$ y agregar este tipo de producto no cambia en nada la función lógica.

$$f(A,B,C) = (A + \bar{B} + C\bar{C})(A + B\bar{B} + C)$$

$$= (A + \bar{B} + C)(A + \bar{B} + C)(A + B + C)(A + \bar{B} + C)$$

Para llegar a este resultado nótese que

$$(A+\bar{B})+C\cdot\bar{C}=[(A+\bar{B})+C)][(A+\bar{B})+\bar{C}]$$

Puesto que el primer y cuarto término están duplicados, al eliminarlos llegamos a la expresión final:

$$f(A,B,C)=(A+\bar{B}+C)(A+\bar{B}+\bar{C})(A+B+C)$$

A cada uno de los factores de la expresión que encontramos se le conoce como ***maxitérmino***.

Tal como en la suma de productos que nos indica cuáles de las combinaciones de las variables hacen que la función sea verdadera, así el producto estándar de las sumas especifica la combinación de variables que hacen que ***f*** sea falsa. Para que ***f*** sea igual a 0 basta que alguno o más de los maxitérminos tenga el valor de 0 y a su vez el maxitérmino sólo podrá ser falso cuando todas las variables que lo componen tengan el valor de falso. Del ejemplo anterior, si analizamos el primero de los maxitérminos observamos que ***f*** es falsa cuando ***A=0***, $\bar{B}=0$ y ***C=0*** o, equivalentemente, ***A=0, B=1*** y ***C=0***.

De la misma forma que con los minitérminos, cada maxitérmino corresponde a un renglón de la tabla de verdad de la función en que la función ***f*** tiene un valor de 0. Por ejemplo, si examinamos la tabla de verdad de una función de tres variables y vemos que en un renglón la función es falsa cuando ***A=0, B=0*** y ***C=0***, entonces hemos encontrado un maxitérmino que corresponde a ***(A+B+C).*** El número de maxitérminos de una función ***f*** es el mismo número de renglones en los que la tabla de verdad toma un valor de 0.

En resumen, una función lógica puede ser expresada como una suma estándar de productos o como un producto estándar de sumas. En el primer caso la función está expresada en minitérminos que especifican cuando la función ***f*** es verdadera. En el segundo caso la función está expresada en maxitérminos que especifican cuando la función es falsa.

Recordando la dualidad existente en las funciones lógicas, podemos ver la correspondencia entre las dos formas de expresar la misma función. Si tenemos una función de cuatro variables en las cuales son posibles 2^4=16 permutaciones con repetición y vemos en su tabla de verdad que 10 de ellas tienen el valor de 1 y por lo tanto tenemos 10 minitérminos, podemos anticipar que si queremos expresar la función ***f*** como maxitérminos deben haber

exactamente 16-10 =6 de tales términos. Si notamos que uno de los maxitérminos es $(A+\bar{B}+\bar{C}+D)$, significa que la función ***f=0***, cuando *A=0*, *B=1*, *C=1* y *D=0*. El correspondiente dual $\bar{A}BC\bar{D}$ nos indicará que la función es igual a 1.

5.2 Especificaciones de Minitérminos y Maxitérminos de una Función

La estandarización introducida en la sección anterior nos permite presentar ahora una forma sintetizada de notación para las funciones lógicas.

Convengamos antes que una función de ***x*** variables siempre será escrita con todas sus variables en los minitérminos o maxitérminos y que éstas deberán estar en orden alfabético. Así, nunca escribiremos ***ADBC*** sino ***ABCD***. Si asignamos el dígito 1 a las variables no complementadas y 0 a las que sí lo están, podemos representar a cada minitérmino de la función lógica como un número en binario o como su equivalente en decimal. Por ejemplo, el minitérmino $\bar{A}BC$ en una función de tres variables equivale al número 011_2 o tres decimales que representaremos como $\boldsymbol{m_3}$. De esta misma forma $\bar{A}\bar{B}C$ equivale a $\boldsymbol{m_1}$ y ***ABC*** a $\boldsymbol{m_7}$.

Para el caso de los maxitérminos la asignación binaria es el dual de los minitérminos, asignaremos un 0 a la variable no complementada y un 1 a la complementada $(A+\bar{B}+C)$ corresponde a 010_2 o 2 decimal representado con $\boldsymbol{M_2}$ y ***(A+B+C)*** es 000_2 o el cero decimal, representado con $\boldsymbol{M_0}$.

Una función lógica puede ser representada de forma muy conveniente con esta nueva notación. Supongamos que una función de tres variables tiene únicamente los minitérminos 0, 2, 4, 5 y 7:

$$f(A,B,C)=m_0+m_2+m_4+m_5+m_7$$

o aún de forma más simple y condensada:

$$f(A,B,C)=\sum m(0,2,4,5,7)$$

de forma similar si se tratase de los maxitérmimos de la misma función (el dual):

$$f(A,B,C)=M_1+M_3+M_6$$

o también, en su forma abreviada:

$$f(A,B,C) = \prod M(1,3,6)$$

Ejemplo

5.5 Representar la siguiente función con la nueva notación abreviada tanto con sus minitérminos como con sus maxitérminos:

$$f(A,B,C) = \bar{A}\bar{B}\bar{C} + \bar{A}BC + A\bar{B}C + ABC$$

Respuesta

Asignando valor a cada una de las variables de los minitérminos de la función tenemos que los términos corresponden a: 000, 011, 101, 111 que en su equivalente decimal es 0, 3, 5, 7:

$$f(A,B,C) = \sum m(0,3,5,7) = \prod M(1,2,4,6)$$

Notemos en el ejemplo anterior que, al ser una función expresada en minitérminos, el dual de la misma expresada en maxitérminos se encuentra aplicando la siguiente regla: si el minitérmino $\boldsymbol{m_k}$ no aparece en la función, el maxitérmino $\boldsymbol{M_k}$ sí aparecerá.

De forma similar supongamos que expresamos una función ***f*** en términos de una combinación de variables que hacen que ésta valga 1 (sea verdadera), esto es, en función de sus minitérminos. Supongamos también que escribimos una segunda función ***g*** relacionada a ***f*** en que cada minitérmino es reemplazado por un mismo número de maxitérmino. Entonces los 0 de ***f*** serían los 1 de ***g*** y viceversa por lo que *g= f*.



Maurice Karnaugh
(1924-)

Físico y matemático estadunidense que trabajó en los laboratorios Bell y la IBM. Introduce los mapas que llevan su nombre como sistema de simplificación de funciones lógicas en 1953 así como varias patentes para la codificación PCM y circuitos lógicos magnéticos.

5.3 Representación de Funciones Lógicas en Mapas de Karnaugh

El ***Mapa de Karnaugh*** (conocido también como mapa K) es un diagrama que provee un área para representar cada renglón de una tabla de verdad. La utilidad de los mapas K está en la particular forma de localizar las áreas que hacen posible simplificar las expresiones complejas utilizando sólo una simple inspección visual. En las siguientes secciones veremos cómo se puede obtener la simplificación.

Un mapa K para el caso de dos variables ***f(A,B)*** se representa en la figura *5.1*. En esa figura se presenta la relación entre un mapa K y la tabla de verdad de una función *f*. La columna del valor de la función ***f*** no está especificada pues por el momento no nos interesa su valor. Los renglones y columnas se han etiquetado de forma decimal y corresponden a la equivalencia del número binario formado por la correspondencia de ***A*** y ***B***. El mapa K tiene 4 celdas que corresponden a los cuatro valores posibles de combinar dos variables (en este caso específico).

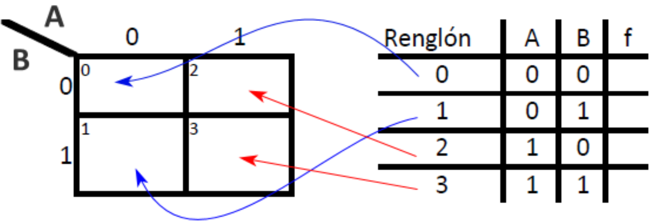


Figura 5.47 Mapa de Karnaugh y Tabla de Verdad.

Este mapa puede usarse ahora como un substituto de la tabla de verdad. Si observamos la figura *5.2a* donde se supone un valor a la función ***f***, se ve que el mapa corresponde a la tabla y que la función se encuentra expresada en minitérminos si escogemos los 1 (la función ***f*** es verdadera), para este caso:

$$f(A,B) = \bar{A}\bar{B} + AB = m_0 + m_3$$

y el caso dual, en maxitérminos; lo vemos cuando escogemos los 0 (la función ***f*** es falsa)

$$f(A,B) = (\bar{A} + B)(A + \bar{B}) = M_1 + M_2$$

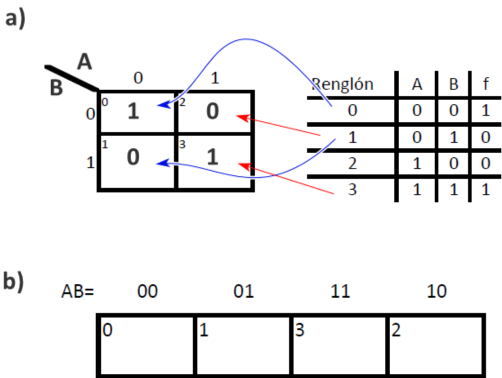


Figura 5.48 Llenando un mapa de Karnaugh.

En el mapa K de la figura *5.2a* vemos, por lo tanto, la representación redundante de minitérminos y maxitérminos. Se escoge representar por sencillez únicamente los minitérminos (unos) o los maxitérminos (ceros) y se entiende que donde hay unos, los ceros no se marcan y viceversa.

Una representación alternativa para mapas K de dos variables se muestra en la figura *5.2b*. Obsérvese que la numeración no es secuencial, sino que sigue el código de Gray descrito en el capítulo 2. La razón de esto se analizará en la siguiente sección.

Analizando los mapas para una función específica nos podemos dar cuenta del porqué del nombre minitérmino o maxitérmino. Considere la función $f(A,B) = A\overline{B}$, esto es, con un sólo minitérmino. El mapa K correspondiente aparece en la figura *5.3a*. Nótese que el minitérmino llena con unos el área mínima del mapa (sólo una celda). Si escogemos la representación de una función por medio de maxitérminos, por ejemplo $f(A,B) = (A + \overline{B})$, los unos llenan el máximo del área del mapa (tres celdas del mapa de la figura *5.3b*).

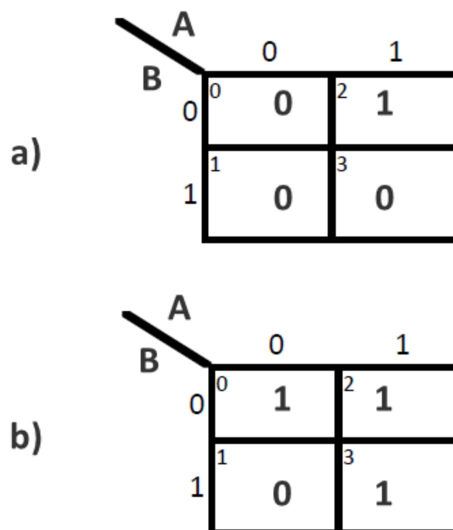


Figura 5.49 Representación con minitérminos y maxitérminos.

5.3.1 Mapa de Karnaugh para Tres y Cuatro Variables

Un mapa tipo K para tres variables se representa en la figura *5.4a*. Una vez más, las celdas están numeradas de acuerdo con los minitérminos (o maxitérminos) que representan. La numeración se hizo de forma tal que la variable ***A*** es el dígito más representativo.

Nótese también, que se sigue la convención del código Gray en la que sólo se permite el cambio de un bit de una celda a la otra.

El mapa para cuatro variables se muestra en la figura *5.4b* y la variable ***A*** corresponde al bit más significativo. Es posible hacer mapas para 5 y 6 variables, pero lo dejaremos para una sección posterior.

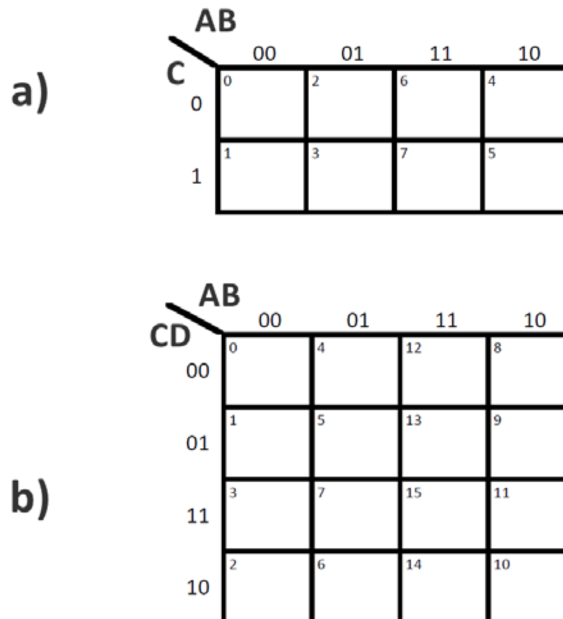


Figura 5.50 Mapa de Karnaugh para 3 y 4 variables.

5.3.2 Simplificación de Funciones con Mapas de Karnaugh

La característica principal de los mapas de Karnaugh es que las celdas que están juntas ya sea horizontal o verticalmente (pero no en diagonal) y que corresponden a minitérminos o maxitérminos, difieren en una sola variable que en un término aparece complementada y en el otro no. Es precisamente para obtener esta combinación que se usa la numeración de celdas en forma del código de Gray y no usando una numeración normal. Para ver los beneficios de esta disposición, analicemos los minitérminos $\boldsymbol{m_8}$ (1000 en binario) y $\boldsymbol{m_{12}}$ (1100 en binario) de una función de cuatro variables que se encuentran en celdas adyacentes en un mapa K:

$$m_8 = A\bar{B}\bar{C}\bar{D}$$

$$m_{12} = AB\bar{C}\bar{D}$$

Nótese que sólo se diferencian en una variable (***B***) que aparece en forma complementada en uno y no complementada en otro. Los términos pueden combinarse para su reducción:

$$A\bar{B}\bar{C}\bar{D} + AB\bar{C}\bar{D} = A\bar{C}\bar{D}(\bar{B} + B) = \bar{A}\bar{C}\bar{D}$$

Así, los dos términos que tenían cuatro variables se han reducido a uno con tres variables. La variable que se encontraba complementada en un término y en el otro no, ha sido eliminada. Si estos minitérminos se encontrasen adyacentes a otros pares, se podrían haber combinado de forma similar para su reducción.

Como principio general tenemos que:

> Cualquier par de celdas adyacentes de minitérminos pueden combinarse para su reducción al analizar la variable que está complementada en uno de ellos y no complementada en el otro.

Directamente del mapa sólo es necesario ver qué variable cambia de 1 a 0 o de 0 a 1 para que ésta se elimine. En el caso anterior del minitérmino m_8 con el m_{12} podemos ver en el mapa que la variable B cambia de 1 (en el minitérmino 12) a 0 (en el minitérmino 8).

En la figura *5.5* observe que los minitérminos que no se encuentran adyacentes pero que, sin embargo, sólo cambian en una variable, son factibles de reducción. En la figura *5.5* realizamos el mapa K de la función:

$$f(A,B,C,D) = \sum m(2,3,8,10,12)$$

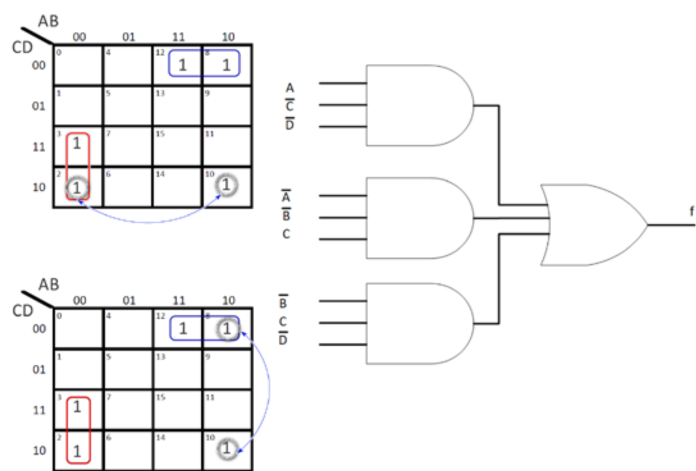


Figura 5.51 Simplificación e implementación de una función.

Se puede reducir de dos formas la función tal como se indica en la figura. Las dos nos dan un mínimo y las dos son aceptables.

Ejercicio

5.1 Simplifique la función de la figura *5.5* escogiendo grupos de minitérminos.

Respuesta

$f(A,B,C,D) = A\bar{C}\bar{D} + \bar{A}\bar{B}C + \bar{B}C\bar{D}$

Del ejercicio anterior (*5.1*) vale la pena mencionar que se puede usar una celda más de una vez para combinar con todas las demás adyacentes y que no se debe dejar ningún minitérmino sin combinar.

En la figura *5.6* mostramos algunos agrupamientos de variables. En general se pueden agrupar 2^n celdas adyacentes (en las que cambie de complementada a no, una sola variable). El caso de 16 celdas adyacentes indica, para cuatro variables, que la función es siempre verdadera independientemente de las variables por lo que generalmente este tipo de agrupamiento sólo es significativo para funciones de 5 variables o más.

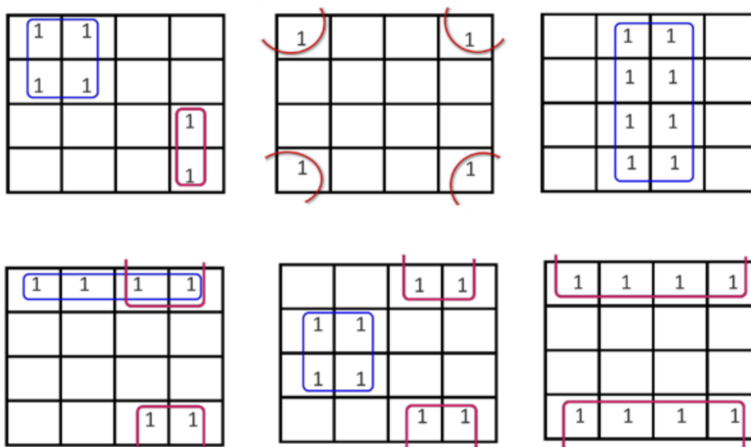


Figura 5.52 Ejemplos de agrupamientos diversos de minitérminos.

5.3.3 Mapas de Karnaugh para Cinco o más Variables

Si seguimos el camino que nos llevó al desarrollo de mapas de 2 a cuatro variables, podemos, por extensión, llegar a los de cinco y seis variables. En la figura *5.7* mostramos dos formas de representar los mapas de cinco variables. El mapa de cinco variables es simplemente uno de cuatro variables duplicado para el caso de la quinta variable complementada y sin complementar. Nuevamente se usa el código Gray reflejado para la numeración. El mapa

preserva la característica de que dos celdas contiguas son susceptibles de reducción y el renglón superior es contiguo al inferior; así como la columna izquierda lo es al de la derecha.

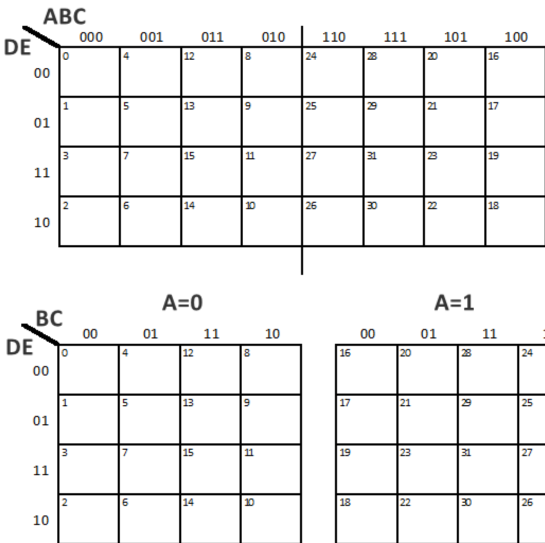


Figura 5.53 Mapas de Karnaugh de 5 variables.

La segunda forma de representar el mapa K de cinco variables logra que sea mucho más fácil la visualización y simplificación de funciones lógicas. Aquí la celda de la sección ***A=0*** es adyacente a su correspondiente en ***A=1***. Por ejemplo, m_5 es adyacente a m_{21}, m_{14} a m_{30}, etc.

Siguiendo las mismas consideraciones que para un mapa de cinco variables llegamos al de seis de la figura *5.8*. Las celdas que se consideran adyacentes son las mismas que en el caso de las cuatro variables; pero también intervienen las de aquellas cajas verticales y horizontales de aquella en que está la celda de interés. Por ejemplo, m_{13} es adyacente a m_{29} y a m_{45}, m_{25} es adyacente a m_9 y a m_{57}, etc.

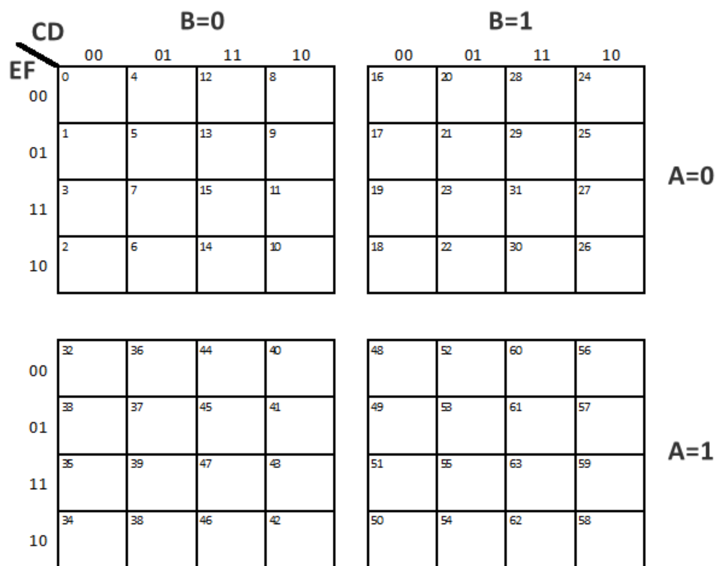


Figura 5.54 Mapa de Karnaugh de 6 variables.

Los mapas de Karnaugh tienen la virtud, como ya lo hemos dicho, de permitir la visualización de términos adyacentes, cuando el número de variables que intervienen en la función crece. Crece también la complejidad de los mapas y se ve reducida su efectividad. Para más de 5 o 6 variables se prefieren otros métodos de simplificación tales como los tabulares. Debido a la dificultad inherente de los mapas K cuando el número de variables crece, restringiremos los ejemplos a funciones de cuatro o menos variables.

5.3.4 El Uso de Mapas de Karnaugh

Como ya hemos visto, una vez que una función lógica se expresa en forma estándar de la suma de productos (minitérminos), el mapa K se puede emplear para simplificar la función aplicando:

- La combinación de celdas (minitérminos) que son seleccionadas deben usarse por lo menos una vez. Nótese que una celda puede usarse más de una vez.
- Las selecciones deben realizarse de forma que incluyan el máximo número de celdas para tener las menos combinaciones posibles.
- Las celdas se deben combinar en potencias de 2 siendo la mínima 2^0=1.

Las combinaciones son llamadas algunas veces ***productos*** y otras ***implicantes primos***. Existen casos en los que puede haber varias combinaciones posibles de los implicantes primos (como el caso

del ejercicio *5.1*) pero algunos de los productos no desaparecen en ninguna de las combinaciones. A estos productos persistentes se les conoce con el nombre de ***implicantes primos esenciales*** y a los que pueden cambiar según se seleccione la simplificación como no esenciales.

Cuando hemos expresado la función como una suma de implicantes primos, por cada implicante involucrado necesitaremos una compuerta tipo ***Y***. El número de entradas a esta compuerta decrece conforme el número de celdas que intervienen en el implicante crece. La economía de la realización de una función por medio de compuertas lógicas se juzga por el número que se emplea. En dos estructuras con el mismo número de compuertas se valúan entonces las entradas a cada una de las compuertas.

Nuestra preocupación por encontrar los implicantes primos en un mapa K con el máximo número de celdas en cada producto nos puede llevar a situaciones ambiguas como la de la figura *5.9* en la que en nuestro afán de simplificar podemos escoger productos de más.

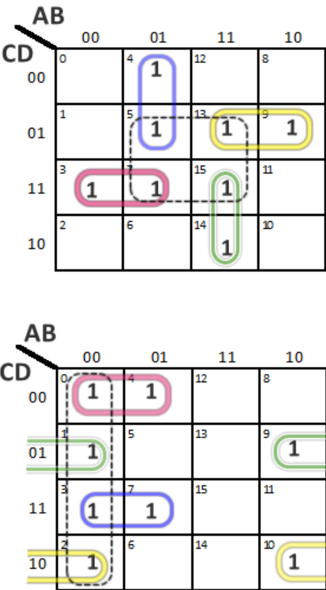


Figura 5.55 Casos ambiguos en la selección de implicantes.

El siguiente algoritmo nos lleva a una mínima expresión para la función lógica y evita los problemas a los que nos referimos anteriormente:

- Encierre y acepte como implicante primo esencial cualquier celda o celdas que no pueden ser combinadas con otras.
- Identifique las celdas que pueden ser combinadas con otra celda única en sólo una forma. Encierre tal combinación de dos celdas. Los grupos que se pueden combinar en más de una forma se ignoran por el momento.
- Identifique las celdas que pueden ser combinadas con otras tres en una sola forma. Si todas las cuatro celdas no están ya implicadas en otra agrupación, encierre al grupo. Una vez más, las celdas que pueden agruparse en grupos de cuatro de más de una forma son omitidas temporalmente.
- Repita el procedimiento para grupos de 8, 16, etc.
- Una vez seleccionados todos los grupos por los procedimientos anteriores, quedan aún algunas agrupaciones sin resolver. Estas celdas antes ignoradas pueden ser combinadas entre ellas o con las otras en una forma arbitraria. Se debe tratar de incluir las celdas sobrantes en tan pocos agrupamientos como sea posible.

Aplicaremos el procedimiento descrito en los siguientes dos ejercicios.

Ejemplos

5.6 Simplifique la función:

$$f(A,B,C,D) = \sum m(0,2,3,4,5,7,8,9,13,15)$$

Respuesta

Una vez encontrado el mapa K de la función procedemos a simplificar y una de las posibles soluciones es:

$$f(A,B,C,D) = \bar{A}\bar{C}\bar{D} + \bar{A}\bar{B}C + A\bar{B}\bar{C} + BD$$

En el siguiente ejemplo, proponemos un problema donde la función está expresada por sus maxitérminos en lugar de sus minitérminos:

5.7 Simplificar la función de cuatro variables

$$f(A,B,C,D) = \prod m(0,3,4,5,6,7,11,13,14,15)$$

Respuesta

El mapa K de la función se muestra en la figura *5.10* junto con una posible agrupación y su realización con compuertas lógicas. La función simplificada queda expresada como:

$$f(A,B,C,D) = (A + C + D)(\bar{C} + \bar{D})(\bar{B} + \bar{D})(\bar{B} + \bar{C})$$

CD \ AB	00	01	11	10
00	0: 0	4: 0	12:	8:
01	1:	5: 0	13: 0	9:
11	3: 0	7: 0	15: 0	11: 0
10	2:	6: 0	14: 0	10:

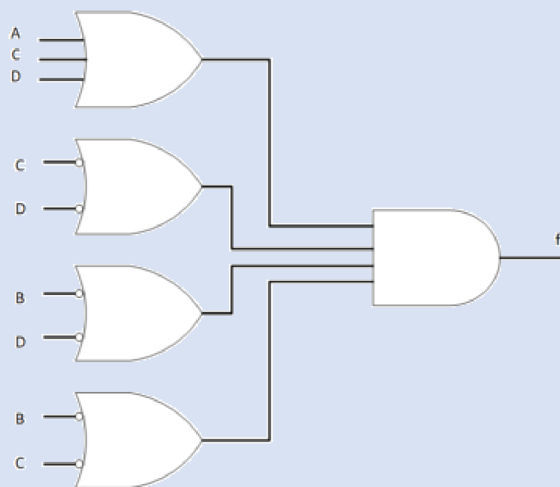


Figura 5.56 Función expresada en maxtérminos y su implementación.

Es importante comparar la realización del ejemplo *5.7* con la encontrada en el ejemplo *5.6* que fue expresada por medio de minitérminos.

5.3.5 Mapas de Funciones no Expresadas en Minitérminos

Toda la presentación hasta el momento supone que la función debe ser expresada en minitérminos o maxitérminos y que, si por alguna causa no es así, debe expandirse hasta lograr que esto suceda. Pero en la práctica, si la función no se encuentra como suma

estándar de los productos o producto estándar de la suma, no es necesario convertirla. Como alternativa se puede insertar cada uno de los términos de la función en un mapa de Karnaugh. Para ejemplificar considérese la función:

$$f(A,B,C,D) = \bar{A}\bar{B}\bar{C}\bar{D} + B\bar{C}D + \bar{A}\bar{C} + A$$

en la que sólo el primer término es un minitérmino (refiérase a la figura *5.11*). El primer término tiene cabida directa en el mapa. El segundo término corresponde a *B=1****,*** *C=0* y *D=1* y es independiente del valor de ***A***, por lo que debe ser marcado para los dos valores de ***A*** y esto corresponde a los minitérminos $\boldsymbol{m_5}$ y $\boldsymbol{m_{13}}$. De forma similar el tercer término corresponde a *A=0* y *C=0* y para ambos valores de ***B*** y ***D***. Finalmente, el último término corresponde a *A=1* y se coloca en todas las celdas donde se cumpla está condición. El resultado de la simplificación nos arroja:

$$f(A,B,C,D) = A + \bar{C}$$

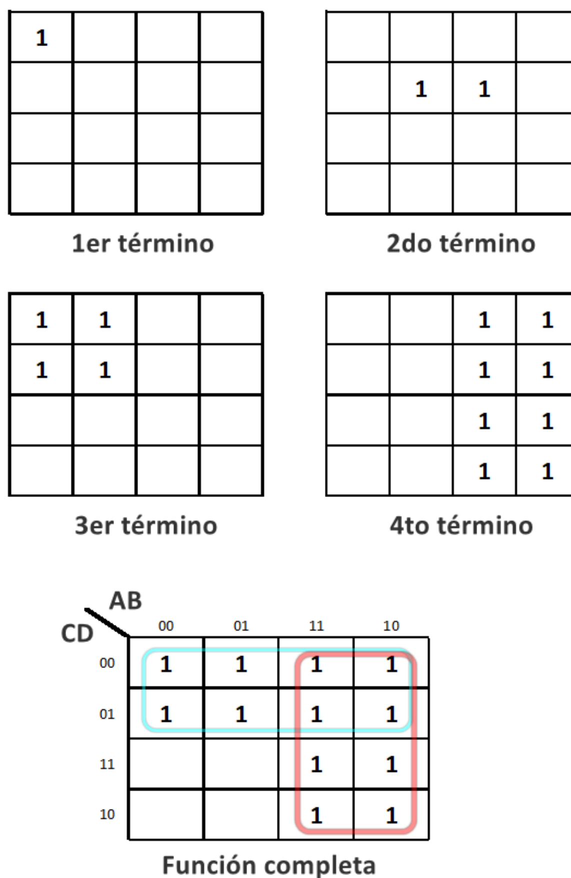


Figura 5.57 Función no expresada en minitérminos.

Ejercicio

5.2 Repita el ejemplo *4.1* simplificando ahora con un mapa de Karnaugh. Se debe obtener primero una tabla de verdad para la función *k(A,B,C,D)* que indique si el desastre puede ocurrir y luego proceder a ubicar en el mapa todos los minitérminos (función *k=1*).

5.4 Síntesis Usando Compuertas NOY y NOO

Al sintetizar funciones lógicas como suma de productos, su realización física implica una serie de compuertas ***Y*** seguidas por una única compuerta ***O*** que reúne a las demás y realiza la suma. Al ser las compuertas ***Y*** las primeras en recibir la señal de las variables, se les denomina como primer nivel; de la misma forma la compuerta ***O*** se conoce como segundo nivel. Por lo que a un sistema similar al de la figura *5.11* se le denomina sistema de dos niveles ***Y—O***. Si la función lógica se expresa como producto de sumas, la situación queda al revés de lo descrito anteriormente; trabajamos con un sistema de dos niveles ***O—Y***. Es importante notar que muchas veces las realizaciones de más de dos niveles pueden ser más sencillas que las de dos niveles, sin embargo, no existe un método sencillo de diseñar a más de dos niveles. Mientras menos niveles de compuertas tengamos en una realización, menor es el retardo de la señal desde su entrada hasta obtener un resultado a la salida.

En el capítulo anterior resaltamos el hecho de que todo circuito puede realizarse enteramente con compuertas del tipo ***NOY*** o ***NOO*** y que existen buenas razones para desear esto. Mostraremos cómo se puede lograr tal efecto con un ejemplo.

Ejemplo

5.8 Considérese una función cuyo mapa K aparece en la figura *5.12a*. Como una suma de productos podemos leerla como:

$$f(A,B,C,D) = \bar{A}B + A\bar{C} \;\; o$$

$$f(A,B,C,D) = (A+B)(\bar{A}+\bar{C})$$

La realización física de dos niveles ***O–Y*** se muestra en la figura *5.12b* y la de dos niveles ***Y–O*** en la *5.12c*. Si se aplica el teorema De Morgan dos veces al producto de las sumas obtendremos:

$$\bar{f} = \overline{\bar{A}B + A\bar{C}} \;=\; \overline{(\bar{A}B)} + \overline{(A\bar{C})}$$

por lo que

$$f = \bar{\bar{f}} = \overline{\overline{(\bar{A}B)} + \overline{(A\bar{C})}}$$

que se muestra en la figura *5.12d* usando sólo compuertas ***NOY***. Observe que el circuito de la figura *5.12d* es igual al de la figura *5.12b* reemplazando toda compuerta por una compuerta ***NOY***.

Si ahora aplicamos el teorema De Morgan dos veces a la suma de los productos obtendremos:

$$\bar{f} = \overline{(A+B)(\bar{A}+\bar{C})} = \overline{(A+B)} + \overline{(\bar{A}+\bar{C})} \;\; y$$

$$f = \bar{\bar{f}} = \overline{\overline{(A+B)} + \overline{(\bar{A}+\bar{C})}}$$

Que, como se muestra en la figura *5.12e*, es similar a la de la figura *5.12c* reemplazando cada compuerta con una del tipo ***NOO***.

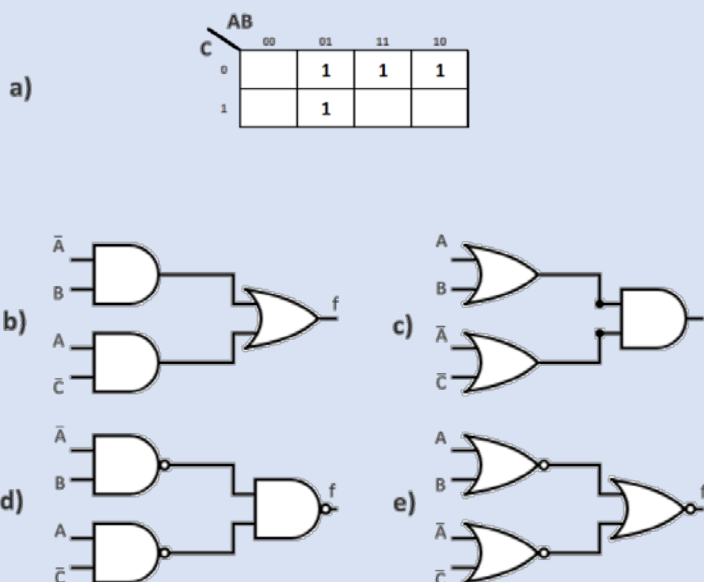


Figura 5.58 Síntesis usando compuertas NOO y NOY.

En resumen:

- Para llegar a un circuito con sólo compuertas tipo ***NOY***, comience por expresar ***f*** como una suma de productos y

luego reemplace toda compuerta resultante por compuertas ***NOY***.

- Para llegar a un circuito con sólo compuertas tipo ***NOO***, comience por expresar ***f*** como un producto de sumas y luego reemplace toda compuerta resultante por compuertas ***NOO***.

5.5 Funciones no Especificadas Completamente

Una función ***f*** se define especificando cada posible combinación de variables ya sea que la función tenga valor ***f=0*** o ***f=1.*** Tal especificación permite llenar un mapa de Karnaugh de forma inmediata para proceder luego a expresar la función en su forma más simple.

Suponga que se tiene una función ***f*** para la cual están especificadas algunas (pero no todas) las combinaciones de variables. En tal caso, un número distinto de funciones son posibles. Todas ellas satisfacen las especificaciones, aunque difirieran una de otra en los valores que no se tienen especificados. La disyuntiva será, ¿Cuál de las funciones escoger, entre las que satisfacen nuestras especificaciones, para llegar a la forma más sencilla?

Tales especificaciones incompletas surgen en la práctica de dos formas:

- No nos importan algunos de los estados.
- Sabemos que ciertas combinaciones de variables no van a suceder. En este caso podemos suponer que no nos importan pues el efecto neto es el mismo.

Para ilustrar el punto considere la siguiente función donde la ***i*** indica estados que no nos importan (ver figura *5.13*):

$$f(A,B,C,D) = \sum m(1,2,5,6,9) + \sum i(10,11,12,13,14,15)$$

En el mapa K marcamos los minitérminos con 1 (como es nuestra costumbre) y las situaciones de no importa con una X. Si escogemos del mapa para su simplificación sólo los minitérminos (valores verdaderos de la función), ignorando las situaciones de no importa, la simplificación queda como:

$$f = \bar{A}\bar{C}D + \bar{B}\bar{C}D = \bar{A}C\bar{D}$$

Pero si usamos los estados de no importa ya sea como 1 o como 0, según nos convenga para la simplificación:

$$f = \bar{C}D + C\bar{D}$$

por lo que el procedimiento a seguir es siempre:

> Utilizar las situaciones de no importa ya sea como ceros o unos según convenga al agrupamiento de 1 especificados por los minitérminos de la función a simplificar.

CD \ AB	00	01	11	10
00			X	
01	1	1	X	1
11			X	X
10	1	1	X	X

Figura 5.59 Funciones no especificadas completamente.

5.6 Otras Técnicas de Reducción

Los mapas de Karnaugh son una técnica invaluable para la reducción de funciones lógicas. Sin embargo, su utilidad queda limitada por los siguientes factores:

- Se trata de un método de prueba y error y no ofrece la menor garantía de producir la mejor realización.
- Depende de la capacidad intuitiva del ser humano para reconocer patrones lo que hace que sea inapropiado para cualquier forma de mecanización.
- Para seis o más variables, es difícil que el diseñador tenga la seguridad de que ha escogido el conjunto de productos más pequeños posible.

Es por esto por lo que otros investigadores han propuesto métodos alternativos como el ***tabular de Quine y McCluskey*** que ayudan a corregir estas deficiencias. El método tabular propuesto por estos dos investigadores garantiza una realización mínima y se puede describir de forma algorítmica, esto es, apropiada para programar en una computadora.

El método es fundamentalmente un procedimiento organizado ingeniosamente para efectuar búsquedas exhaustivas de todas las combinaciones posibles de minitérminos.

Algoritmo Quine-McCluskey (QMC)

También conocido como el método de los principales implicantes, es un método utilizado para minimizar las funciones booleanas. Fue desarrollado por Willard V. Quine en 1952 y ampliado por Edward J. McCluskey en 1956. Es funcionalmente idéntico al mapeo de Karnaugh, pero la forma tabular lo hace más eficiente para su uso en algoritmos informáticos y también ofrece una forma determinista de verificar que se haya alcanzado la forma mínima de una función booleana. Se le conoce a veces como el método de tabulación.

Se deja como ejercicio documentar y de ser posible realizar un algoritmo en pseudocódigo del algoritmo de Quine-McCluskey.

5.7 Resumen

Los minitérminos y maxitérminos proveen una forma de sintetizar una función lógica y dan pie a varias técnicas de reducción que no emplean los teoremas de la lógica.

Los mapas de Karnaugh son una herramienta importante, aunque no exclusiva, de reducir funciones de hasta seis variables con relativa facilidad de forma visual.

Se deben aprovechar todos los casos en que las funciones no están completamente especificadas para reducir aún más la función.

Usaremos extensivamente esta técnica en los siguientes capítulos cuando necesitemos reducir una función.

5.7.1 Puntos Importantes del Capítulo

- El mapa de Karnaugh (mapa K) es una forma alternativa de representar una función.
- El mapa K usa el código Grey para la simplificación.
- Se deben agrupar tantos pares de unos (o ceros) como sea posible para hacer la simplificación máxima en un mapa K.
- Existen otras técnicas numéricas que deben ser investigadas para sistematizar aún más la simplificación.

5.8 Problemas

5.1 Reduzca las expresiones siguientes a una suma mínima de productos.

a) $(A+\bar{B}+C)(\bar{A}+\bar{B}+\bar{C})$

b) $(A+B)(\bar{B}+\bar{A})$

c) $AB(C+D)E+(C+D)AC$

5.2 Escriba las funciones del problema *5.1* como:

a) Una suma estándar de productos.

b) Un producto estándar de sumas.

5.3 Simplifique:

a) $f(A,B,C,D) = \sum m(0,2,3,4,10,12)$

b) $f(A,B,C,D) = \sum m(2,4,5,6,12,14)$

c) $f(A,B,C,D) = \sum m(0,2,3,4,12)$

d) $f(A,B,C,D) = \prod M(0,2,3,4,8,10,12)$

e) $f(A,B,C) = \prod M(0,1,2,4)$

f) $f(A,B,C,D) = \prod M(0,8,10,12)$

5.4 Usando el mínimo número de compuertas:

a) ***NOY*** de dos entradas.

b) ***NOY*** de tres entradas.

c) ***NOO*** de dos entradas.

d) ***NOO*** de tres entradas.

simplifique e implemente las siguientes funciones:

a) $f(A,B,C,D) = \sum m(0,1,4,5,9,11,16) + \sum i(10,13)$

b) $f(A,B,C,D) = \sum m(0,13,14,15) + \sum i(8,9,11)$

5.5 Diseñe un circuito que tenga como entrada 4 líneas de código ***BCD*** (descrito en el capítulo 3) y como salida 7 líneas que manejarán un despliegue digital de 7 segmentos (descrito en capítulo *2*, figura *2.9*). Para comenzar el diseño considere si la línea de salida debe prenderse (1 lógico) o no de acuerdo con el número decimal y a la representación visual de este número en el despliegue. Realice posteriormente la simplificación con mapas K de cada una de las 7 líneas. El circuito así realizado puede comprarse comercialmente y es llamado ***BCD a 7 segmentos***.

6 Familias Lógicas

Los sistemas digitales usualmente son construidos utilizando los elementos que hemos analizado en capítulos anteriores, tales como transistores, resistencias, diodos, etc. A tales sistemas digitales se les llama ***compuertas lógicas***. Un ejemplo de compuerta típica podría ser el circuito inversor que ya describimos al analizar los transistores. Un circuito lógico combina las entradas de acuerdo con ciertas reglas que definen la función de la compuerta. Pueden incluirse etapas adicionales en la compuerta ya sea para aumentar su velocidad de respuesta, mejorar la forma de onda de la salida, etc. A través de los años se han utilizado varias configuraciones genéricas de circuitos para construir las compuertas lógicas. Estas constituyen las llamadas ***familias lógicas*** de circuitos integrados, por ejemplo, entre las familias más importantes contamos con la ***TLL*** (lógica de transistor a transistor, abreviada también T^2L), familia que se caracteriza por el uso de transistores tanto en la etapa de entrada como en las subsecuentes etapas de amplificación y salida. Los circuitos de una familia más vieja y ahora obsoleta, llamada ***DTL*** (lógica de diodo–transistor) usa diodos en lugar de transistores en la etapa de entrada. Además de compartir una estructura común de circuitos electrónicos, los miembros de las familias son compatibles con los otros. La compatibilidad, usada en este contexto, significa el uso de la misma corriente y voltaje en los rangos de señales para representar valores lógicos y generar las señales de salida que pueden ser directamente conectadas a las líneas de entrada de otros miembros de la misma familia. Las diferentes familias pueden ser incompatibles en más de una forma, necesitando el uso de circuitos especiales llamados de ***interfaz*** para ligar miembros de distintas familias en un circuito común.

Una familia ideal de circuitos integrados combinará:

- Operación a alta velocidad.
- Baja potencia de consumo.
- Bajo coste de producción.
- Facilidad de su uso en el diseño de sistemas.

La existencia de numerosas familias lógicas incompatibles surge del hecho de que todas las familias lógicas prácticas se desvían de una u otra forma del ideal buscado. Por ejemplo, las familias

lógicas llamadas bipolares son relativamente rápidas, pero también tienen un gran consumo de potencia. Algunas familias llamadas ***MOS*** (Semiconductores de óxido metálico) tienen muy poco consumo de potencia, pero tienden a ser más lentas que los circuitos bipolares que desempeñan la misma función. El diseñador de sistemas digitales se enfrenta, por lo tanto, a distintas familias de circuitos integrados cuyo uso involucra distintos aspectos a considerarse con respecto a su velocidad, consumo y otros factores de diseño.

En la figura *6.1* listamos algunas de las familias más importantes de circuitos integrados que pueden ser de interés para el que diseña o estudia sistemas digitales. Cada circuito integrado bipolar emplea ya sea resistencias tal como en la familia ***RTL*** (Lógica de resistencia a transistor) o diodos, tal como en la familia ***DTL*** para formar combinaciones lógicas de sus señales de entradas. Casi todas estas familias han sido substituidas de una u otra forma por las dos tecnologías dominantes por el momento: ***TTL*** y ***CMOS*** (Semiconductores de óxido metálico complementario). La familia ***TTL*** se forma por transistores, incluyendo algunos con dos o más emisores, que sirven tanto para operaciones lógicas como para amplificaciones de señal. Muchos de los circuitos de integración a baja y mediana escala (***SSI*** y ***MSI***) actualmente se diseñan usando la tecnología ***TTL***.

Otro miembro de la familia bipolar es la tecnología ***ECL*** (lógica acoplada por emisores) que usa una estructura formada por transistores con tiempos más cortos de respuesta que la familia ***TTL,*** pero con un consumo mayor de potencia. Esta familia se usa para construir computadoras grandes y rápidas, pero el consumo de energía, que requiere enfriamiento especial, limita su uso en las microcomputadoras.

La última familia bipolar, ***I^2L*** (lógica de inyección integrada) es quizá la más próxima a la familia ideal. Sin embargo, la tecnología para la manufactura de la familia es compleja y relativamente inmadura por lo que no se usan extensivamente hoy en día.

De la rama ***MOS*** (semiconductores de óxido metálico) de las familias tenemos tres grandes categorías: las familias ***pMOS***, ***nMOS*** y ***CMOS***. La familia ***pMOS*** usa transistores de efecto de campo tipo ***p*** mientras que la familia ***nMOS*** usa transistores de efecto de campo tipo ***n***. Los circuitos ***pMOS*** son algo más sencillos de fabricar que los ***nMOS*** pero más lentos que los segundos porque los

huecos usados como cargas portadoras en los transistores ***pMOS*** tienen menos movilidad que los electrones usados en los transistores tipo ***nMOS***. La familia ***CMOS*** combina tanto transistores ***pMOS*** como ***nMOS*** en aproximadamente el mismo número de forma que tiene un consumo muy bajo de corriente. Todas las familias ***MOS*** son usadas en la fabricación de microprocesadores.

Nuevas variantes de cada familia surgen constantemente para mejorar algunas características de la familia, por ejemplo, la familia ***TTL*** Schottky que añade diodos tipo Schottky para mejorar el tiempo de respuesta.

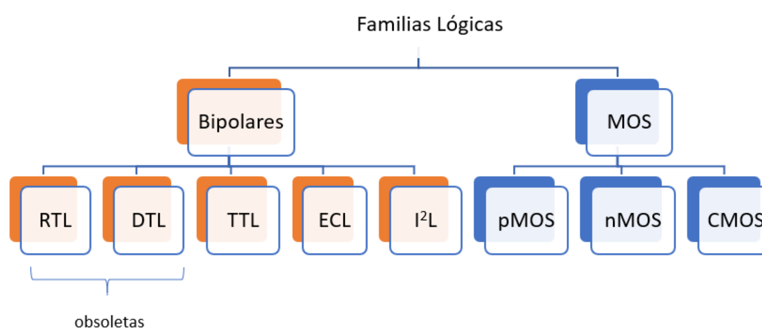


Figura 6.60 Familias lógicas.

6.1 Familias en Desuso

Analizaremos dos familias que, aunque ya no están en uso, nos ayudan a comprender la realización de compuertas lógicas prácticas.

6.1.1 Lógica de Resistencia Transistor

Como su nombre lo implica, esta familia lógica se compone exclusivamente por resistencias y transistores. Hemos analizado ya el funcionamiento de un inversor al estudiar el transistor y el circuito que se muestra en la figura *6.2a* que corresponde a una compuerta ***Y*** (***AND***) de dos entradas implementada en la lógica ***DRL*** (Diodo Resistencia; en desuso). En la figura 6.2b se muestra la implementación de la compuerta ***O*** (***OR***) de 2 entradas de la misma familia.

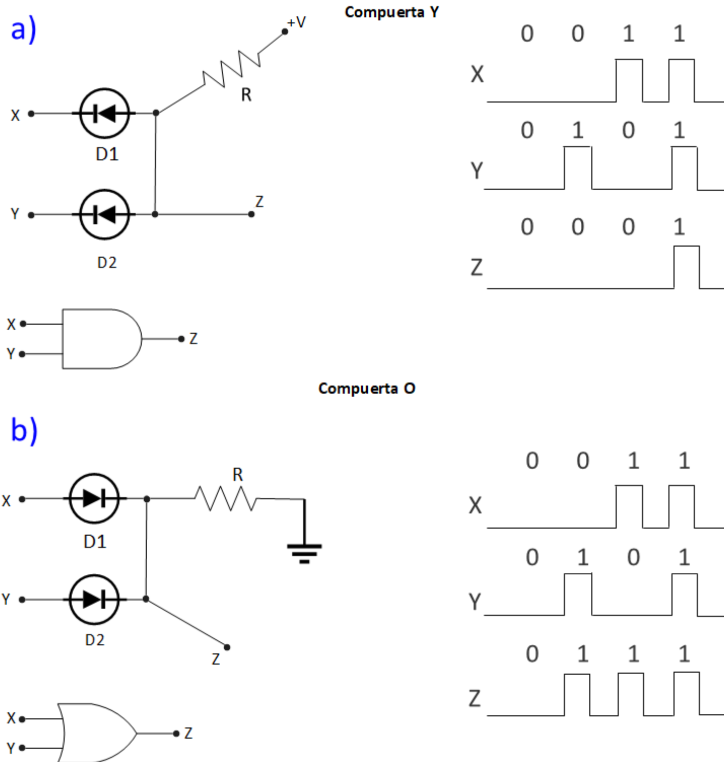


Figura 6.61 Compuerta O e Y de la familia DRL.

Describimos su funcionamiento pues su comprensión facilita la de las otras familias. Refiérase a la figura *6.2a*.

Si cualquiera de las entradas está ***BAJA***, el diodo correspondiente se encuentra polarizado en su zona directa (conduce) y se comporta como un interruptor cerrado. Despreciamos la resistencia directa del diodo y la caída de voltaje a través del diodo. Por lo tanto, el voltaje de salida ***Z*** es de 0V.

Para que la salida ***Z*** sea ***ALTA***, ambas entradas, tanto **X** como ***Y***, tendrán que ser ***ALTAS***. De esa forma los dos diodos ***D₁*** y ***D₂*** actúan como interruptores abiertos dejando pasar la corriente de la fuente a través de la resistencia hacia el nodo ***Z***.

Hemos implementado, así, la función ***Y*** (***AND***). Aunque el circuito funciona, tiene muchas desventajas, en particular a que su salida BAJA la mantiene un diodo, por lo que no se pueden unir muchos de estos dispositivos en línea y que su velocidad de conmutación es lenta debido a la resistencia que "levanta" el voltaje.

Para paliar estas y muchas otras deficiencias de esta tecnología, se pensó substituir los diodos por transistores dando nacimiento

a la familia ***RTL***. En la figura *6.3* representamos un inversor y una compuerta ***NAND*** de la familia ***RTL***.

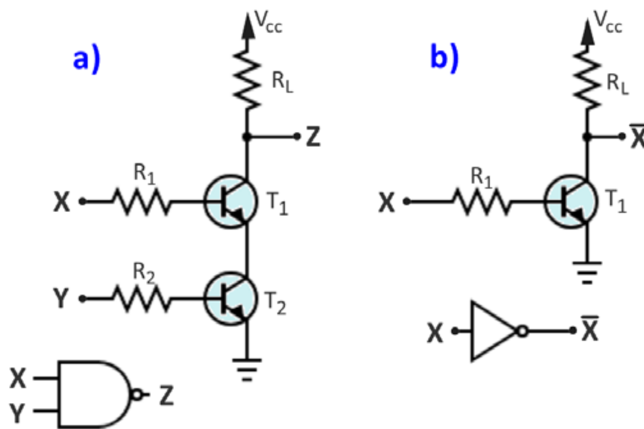


Figura 6.62 Inversor y compuerta NAND de la familia RTR.

El circuito ***NAND*** representado funciona de la siguiente forma: Cuando la entrada ***X*** tiene voltaje, una corriente fluye por R_1 que prende al transistor T_1. El voltaje de entrada debe ser lo suficientemente alto como para proporcionar la corriente suficiente a la base del transistor que permita llevarlo a su estado de saturación. De la misma forma, la resistencia de la base debe ser suficientemente alta para evitar lazos de corriente. Si cualquiera de las entradas es ***ALTA***, la corriente circula por la resistencia R_L hacia tierra del transistor en saturación, lo que causa que la salida sea *BAJA*.

El nivel de salida del circuito depende de la carga resistiva efectiva conectada a su salida. La carga resistiva depende a su vez de qué tantos circuitos se pueden conectar entre sí. A esto se le denomina ***manejo de salida*** (Fan−Out en inglés) que se define como el número de entradas a otras compuertas que son manejadas por la salida de un circuito determinado. La dependencia de la temperatura y los valores absolutos de tolerancia de las resistencias limita el manejo de salida práctico de la familia ***RTL*** a 5 compuertas.

A la vez, el ***manejo de entrada*** (Fan−In en inglés) es el máximo número de entradas que una compuerta puede soportar sin que su funcionalidad se vea afectada.

Las compuertas de la familia ***RTL*** sufren de varias anomalías:

- Alto consumo de corriente.
- Baja inmunidad al ruido.

- Bajo manejo de salida (Fan−Out).
- Bajo manejo de entrada (Fan−In).
- Creación de lazos indeseables de corriente.

Sin embargo, fueron de las primeras familias lógicas y disfrutaron de mucha popularidad por su sencillez y bajo costo.

6.1.2 Lógica de Diodo Transistor

Esta familia evita algunas de las características indeseables de la familia ***RTL*** substituyendo algunas de las resistencias por diodos.

Si observamos la figura *6.4a*, notaremos la simplicidad de la familia ***DTL***. La compuerta que se describe es una del tipo ***NOY***. Para que los diodos conduzcan es necesario que sus entradas (***X*** y ***Y***) estén a tierra. Cualquiera de las dos que se encuentre en esta condición evitará que la salida sea un 1 lógico. En la figura *6.4* se muestra una compuerta del tipo ***NOY*** y una ***NOO*** fabricadas con esta tecnología.

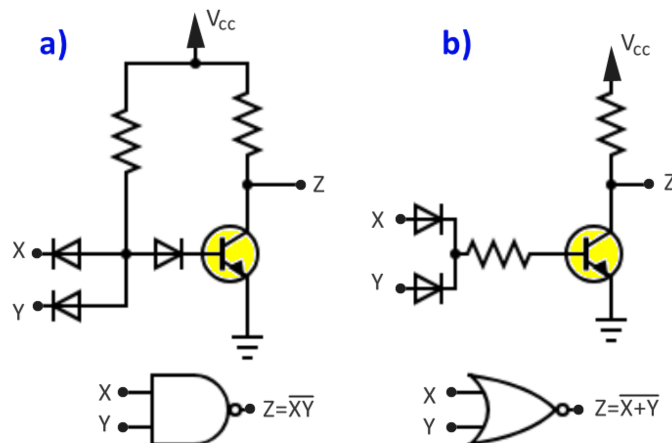


Figura 6.63 Compuertas NOY y NOO de la familia DTL.

Los circuitos ***DTL*** se prestan mejor a la fabricación de circuitos integrados, reduciendo el área y aumentando la impedancia de entrada.

Una versión modificada de esta familia se obtiene reemplazando alguno de los diodos por transistores, creando así una nueva familia llamada ***TLL***. De esta forma, la ganancia que el transistor ofrece se usa para reducir el consumo de potencia de la compuerta y mejorar el manejo de salida (fan−out) de la misma.

La familia ***DTL*** quedó en desuso al introducirse la familia ***TTL*** que mejora aún más el área de integración, la inmunidad al ruido, el consumo de potencia y otras características.

Con la mejora de los procesos de integración se logra pasar de unos cuantos componentes (integración a baja escala o ***SSI***) a varios miles (integración a mediana escala o ***MSI***) y finalmente a millones (integración de alta escala o ***LSI*** e integración a muy alta escala o ***VHI***) pudiéndose emplear otras técnicas, que aunque usen más componentes sean, también, más eficientes.

6.2 Lógica de Transistor a Transistor

Durante cada era de la historia de la tecnología, surge una clase de dispositivos tan versátiles, económicos y confiables que pronto son conocidos como los caballos de batalla. En esta época de circuitos integrados, y desde 1964, los circuitos del tipo ***TTL*** se han ganado este lugar.

Las compuertas ***TTL*** se caracterizan por el uso de dos o más etapas de transistores que desempeñan las operaciones lógicas y la amplificación de la señal. La familia tiene un tiempo de respuesta relativamente rápido de alrededor de 10ns (10×10^{-9} segundos) o menos, aunque su consumo puede llegar a ser alto, de hasta 10mW (10×10^{-3} Watts) por compuerta. Como veremos más adelante hay varias subfamilias ***TTL*** que realizan compromisos tanto de velocidad como de consumo en varias formas. Las compuertas tienden a usar más área que la familia ***MOS*** equivalente y junto con su alto requerimiento de potencia, tienden a limitar su uso en circuitos de alta y muy alta integración (***LSI*** y ***VLSI***).

Si nos referimos a la figura *6.5a* (compuerta del tipo ***NOY***) notaremos que las entradas se realizan por un único transistor que consta de dos emisores. Esto es un aspecto común a la familia ***TTL***. Tales transistores son fáciles de fabricar pues sólo se añade un emisor extra en la estructura básica.

TLL

Circuitos integrados inventados en 1961 por James L. Buie que trabajaba para TRW. Su nombre original era TCTL o lógica acoplada de transistor a transistor. Los primeros dispositivos los fabricó Sylvania en 1963 y fueron usados en el misil Phoenix. Los circuitos se volvieron populares cuando Texas Instruments introdujo su serie 5400 en 1964 y posteriormente la 7400 en 1966.

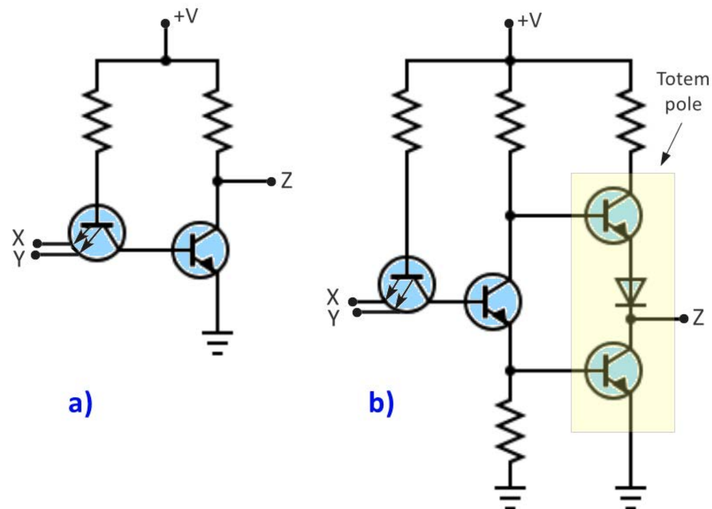


Figura 6.64 Compuertas NOY de la familia TTL.

El desempeño de la familia ***TTL*** tiene varias deficiencias. Supóngase, por ejemplo, que es necesario manejar una carga que tenga una gran capacitancia de entrada. Cuando el transistor de salida se apaga, implicando que la salida sea un 1 lógico, una gran corriente necesita fluir de la fuente por la resistencia del colector hacia la carga capacitiva. El tiempo de carga y descarga del capacitor es directamente proporcional a los valores de la resistencia y la capacitancia. Puesto que la resistencia debe tener de por sí un valor alto, la corriente que podemos hacer pasar por ella disminuye. Esto limita el número de compuertas que pueden conectarse a la salida del circuito y por lo tanto su manejo de salida (fan−out) se limita considerablemente.

El desempeño de los circuitos comerciales se aumenta añadiendo otra etapa de amplificación que aumenta a su vez la capacidad de manejo de salida (ver figura *6.5b*). A esta configuración se le conoce como ***tótem*** (totem pole). Nótese que tanto cuando *Z=1* como cuando *Z=0* el manejo de salida de la compuerta es alto. Si *Z=1* la compuerta funciona como una fuente de corriente y si *Z=0*, como un sumidero de corriente.

Debido a su popularidad, muchos circuitos comunes están disponibles en la familia ***TTL*** de distintas fuentes de fabricantes. Hay un grupo importante de circuitos conocidos como serie 7400, que incluye cientos de circuitos distintos con números estándares de partes, todos ellos comenzando con los números 74 (convención original introducida por la compañía Texas Instrument). La designación 74 indica compuertas de grado comercial que soportan

temperaturas entre 0ºC y 70ºC. Existe una serie conocida como 54 que corresponde a la 74 y es de grado militar soportando temperaturas entre -55ºC y 125ºC. Toda la serie 7400 funciona con una sola fuente de voltaje de 5 Voltios. Los voltajes en el rango de 2 a 5 volts representan el uno lógico y los de 0 a 0.8 volts el cero lógico (ver las figuras *6.7* y *6.8*). La mayoría de los circuitos ***TTL*** vienen en presentación ***DIP*** o planar, sus miembros comparten los mismos requerimientos de entrada y salida y son compatibles los unos con los otros. Los listados completos de especificaciones y los circuitos disponibles se encuentran en los catálogos de los fabricantes de circuitos integrados siendo los mayores Texas Instrument Inc. y National Semiconductors.

Además de la familia "estándar" ***TTL*** existen otra serie de subfamilias disponibles que difieren en velocidad, consumo u otras consideraciones. Por ejemplo, la serie 54H/75H es de mayor velocidad de respuesta mientras que la serie 74L/54L tiene más bajo consumo de potencia. En la familia ***Schottky TTL*** se agregan diodos tipo Schottky entre la base y el colector de la mayoría de los transistores de una compuerta normal ***TTL*** lo que causa la disminución de los tiempos de carga y descarga del transistor asociado, a expensas de más consumo de corriente. El tiempo típico de respuesta disminuye de 10nS a 3nS. Una variante llamada ***TTL Schottky de bajo consumo*** usa distintos valores de resistencia para disminuir el consumo de energía; desafortunadamente se pierden las características de rapidez

6.3 Lógica de Semiconductor de Óxido Metálico

Mientras que varias familias ***TTL*** son usadas ampliamente para la producción de circuitos integrados de baja y mediana escala (***SSI*** y ***MSI***), la tecnología ***MOS*** se prefiere para alta y muy alta integración (***LSI*** y ***VLSI***).

El pequeño tamaño y relativa sencillez de los dispositivos semiconductores de óxido metálico (***MOS***) los hacen muy atractivos para su uso en circuitos digitales. Estos dispositivos de conocen también como transistores de compuerta aislada de efecto de campo (Insulated-Gate Field-Effect transistor o ***IGFET***).

CMOS

El *MOSFET* fue inventado por Mohamed Atalla y Dawon Kahng trabajando para los Laboratorios Bell. Fabricaron el primer dispositivo funcional en noviembre de 1959. La primera familia lógica de circuitos integrados CMOS la introdujo en 1968 RCA como la serie *CD4000COS/MOS*.

Los circuitos ***MOS*** tienen tres ventajas significativas sobre sus contrapartes bipolares:

- Alta densidad de componentes
- Baja disipación de potencia (consumo)
- Alta capacidad de manejo de salida (fan-out)

Sin embargo, sufren de baja velocidad de respuesta, bajo manejo de corriente de salida y requieren de dos fuentes de poder de distintos voltajes para un funcionamiento adecuado.

Debido a su sencillez y facilidad de fabricación, el transistor del tipo ***pMOS*** es el de más uso en esta familia lógica. En la figura *6.6* mostramos una compuerta de dos entradas ***NOY*** (***NAND***).

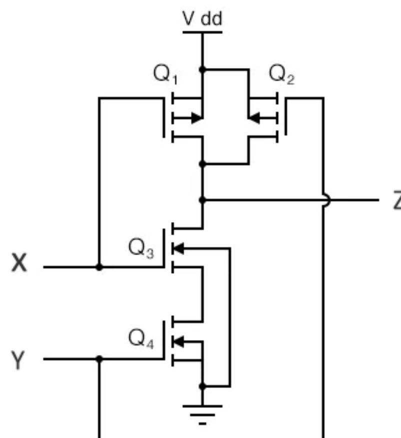


Figura 6.65 Compuerta NOY de la familia MOS.

El uso de transistores del tipo ***pMOS*** y ***nMOS*** en un mismo substrato de silicio nos lleva a la tecnología ***CMOS*** que mejora mucho en los aspectos de bajo consumo de corriente (hasta 100 veces menos que su equivalente ***TTL***) y en su velocidad de respuesta.

La estructura de los transistores ***MOS*** es tal, que el electrodo de la puerta y el substrato donde se construye, se comportan como las placas de un capacitor. Así, un circuito lógico ***MOS*** representa una gran capacitancia a la señal aplicada a sus líneas de entrada. La velocidad con que este capacitor se carga y descarga limita la velocidad de funcionamiento del circuito. La capacitancia incidental formada en el transistor tiene también la propiedad de poder guardar "recordando" por un corto tiempo el voltaje de entrada, aunque éste se remueva temporalmente. Esta característica se emplea en un tipo de circuitos llamados ***dinámicos*** en los que cada circuito individual se desconecta por cortos periodos de la fuente

de poder por medio de señales de control periódicas llamadas señales de ***reloj***. La señal es almacenada en las capacitancias de los circuitos hasta que la fuente se vuelve a conectar. Antes de remover la energía, los capacitores se cargan completamente por medio de una señal llamada de ***refresco***. Desconectando continuamente la fuente de poder permite ahorros en el consumo de energía y lleva a circuitos más sencillos en ciertos casos. Los circuitos lógicos que no controlan las conexiones de la fuente de poder, en la forma descrita anteriormente, se les conocen como ***estáticos***.

Hasta hace poco una gran desventaja era que este tipo de familia usaba dos fuentes de voltaje, típicamente de 12 y −12 voltios. Actualmente se fabrican circuitos compatibles con los niveles ***TTL*** de +5 Volts por lo que las familias pueden mezclarse sin ningún problema ni interfaz.

Así como en la familia ***TTL*** encontramos series de circuitos predeterminados para realizar una función, en la familia ***MOS*** se tiene la serie 4000 (especificación original de RCA, pero en uso por todos los otros fabricantes de estos circuitos). Desgraciadamente la especificación numérica no corresponde a la de la familia ***TTL***. Por ejemplo, el circuito 4011 ***MOS*** (cuatro compuertas de dos entradas del tipo ***NOO***) corresponde al número 7400 de la serie ***TTL***. Últimamente se fabrican los equivalentes de la familia ***TTL*** con tecnología ***CMOS*** y cuya numeración sí corresponde a la serie 74C.

La familia ***CMOS*** tiene las siguientes características:

- Se construye utilizando tanto transistores tipo ***nMOS*** como ***pMOS***.
- Para realizar funciones lógicas usa indistintamente tanto transistores tipo ***p*** como ***n***.
- Tiene un bajo consumo de potencia.
- Su manejo de salida (fan−out) excede al de las otras familias.
- Por el momento se considera como la familia más fiable.

Dejamos al lector interesado la investigación más a fondo de estos temas en la bibliografía sugerida.

6.3.1 Versiones Mejoradas

Como ya mencionado anteriormente, entre las características más importantes de un circuito en general y digital en particular, encontramos:

- La velocidad. Rapidez de respuesta de las salidas de un circuito a cualquier cambio en sus entradas.
- El consumo de potencia. Cantidad de corriente o de potencia que consume el circuito en operación.
- La inmunidad al ruido. Mide la sensibilidad de un circuito al ruido electromagnético ambiental.
- La fiabilidad. Periodo útil de servicio de un circuito, es decir, cuánto tiempo se espera que trabaje antes de que falle.

Los diseños iniciales de circuito integrados ***TTL*** requerían una fuente de poder de 5V, pero la tecnología ***CMOS*** permitía el uso de fuentes entre 3 y 15V. Al bajar los requerimientos de voltaje se reduce la carga almacenada en cualquier capacitor que forme parte del circuito y, por lo tanto, se reduce el tiempo de una transición lógica. Una reducción de energía implica menos disipación de calor. La energía almacenada en un capacitor ***C*** con un voltaje ***V*** que cambia, se define por la fórmula $e = {}^{1}/_{2}\,CV^2$. Al disminuir el voltaje de la fuente de poder de 5V a 3.3V, la potencia se reduce en casi un 60% siendo que la disipación es proporcional al cuadrado del voltaje del suministro.

Supongamos que la mamá de Juan le pide que vaya por agua a un pozo que se encuentra a 5km de distancia, Juan cumple la orden, pero se encuentra con una fuente de agua a 3.3km. En lugar de tardar las 2h que habitualmente tarda en esa tarea, resulta que ahora sólo toma 1h y se cansa menos. Si extrapolamos este ejemplo a un sistema lógico, vemos el resultado de bajar los niveles de voltaje lógicos de “0” y “1” a lo mínimo posible: el circuito aumenta en rapidez de conmutación pues el paso de verdadero a falso y viceversa tarda mucho menos.

Considerando esta óptica se desarrollan lógicas tales como:

- ***HC*** (High Speed *CMOS*; ***CMOS*** de alta velocidad). Versión mejorada de la familia ***CMOS*** en la que su velocidad de conmutación se mejora por un factor de diez. Su manejo de salida (fan-out) también excede al de la familia ***CMOS***.

- ***HCT*** (High Speed *CMOS* with ***TLL*** logic voltages; ***CMOS*** de alta velocidad con voltajes de lógica ***TLL***). Similar a HC a la vez que mantiene compatibilidad con los niveles de voltaje de la familia lógica ***TLL***.
- ***LS-TTL*** (Low−power Schottky ***TLL***; ***TLL*** Schottky de bajo consumo). Usa los altos valores de resistencia de ***TLL*** y los diodos Schottky para aumentar la velocidad y reducir el consumo de la familia ***TLL***. Reemplaza a las subfamilias *H*, *L* y *S* de la familia ***TLL***.

Con la lógica ***HC***, ***HCT*** y ***LS-TTL*** en el mercado, pronto se hizo claro que se requerían mejoras para crear una familia ideal en la que se combinaran alta velocidad, poca disipación de energía y compatibilidad con las viejas familias existentes. Toda una nueva gama de familias lógicas emergió de la tecnología ***CMOS***, la más fiable y adaptada a los criterios requeridos. Entre las designaciones de estas nuevas familias podemos encontrar:

- Lógica ***LV*** (Low Voltage; baja tensión de alimentación)
- Lógica ***LVT*** (baja tensión de alimentación, pero manteniendo compatibilidad con los niveles lógicos de ***TLL***)
- Lógica ***ALVT*** (una versión “avanzada” de la lógica *LVT*)

Existen muchas otras, por ejemplo: *AC/ACT*, *AHC/AHCT*, *ALVC*, *AUC*, *AVC*, *CBT*, *CBTLV*, *FCT* y *LVC* (*LVCMOS*).

Dejamos como ejercicio al lector averiguar las características de estas nuevas lógicas consultando Internet o la bibliografía citada.

6.4 Consideraciones de Carga de la Familia TTL

Los valores lógicos binarios utilizados en sistemas digitales son representados por dos rangos de voltajes. Hemos escogido representar a los voltajes con sus siglas en inglés para tener consistencia con otros libros y textos, así como con las hojas de especificaciones y catálogos de componentes:

- *H* high o alto,
- ***L*** low o bajo,
- ***I*** input o entrada,
- ***O*** output o salida,
- V_{cc} voltaje de alimentación
- V_H ***u***n rango de voltajes altos que simboliza el 1 lógico y
- V_L un rango de voltajes bajos que representa el 0 lógico.

Usaremos la siguiente representación para los voltajes (corrientes) que abordaremos:

- V_H = Voltaje que representa el uno lógico.
- V_L = Voltaje que representa un cero lógico.
- V_{IL} = Nivel de voltaje requerido para un cero lógico a la entrada.
  Garantizado en un máximo de 0.8V.
- V_{IH} = Nivel de voltaje requerido para un uno lógico a la entrada.
  Garantizado en un mínimo de 2.0V.
- V_{OL} = Nivel de voltaje requerido para un cero lógico a la salida.
  Garantizado en un máximo de 0.4V.
- V_{OH} = Nivel de voltaje requerido para un cero lógico a la salida.
  Garantizado en un mínimo de 2.4V.
- V_T = Voltaje de límite (threshold) donde los voltajes de entrada y salida son iguales.

En la figura *6.7a* mostramos los valores de voltajes usados en los circuitos estándar tipo ***TLL***.; otras familias también usan estos voltajes o similares. Cualquier voltaje entre 0.0 y 0.8V denota un cero lógico, mientras que cualquier voltaje comprendido entre 2.0 y 5.0V denota un 1 lógico. El rango intermedio entre 0.8 y 2.0V debe ocurrir solamente cuando una señal está cambiando de un valor lógico a otro. Valores estables de voltaje que caigan en esta región intermedia no son permitidos en el diseño lógico pues no corresponden a ninguno de los dos estados posibles y su efecto en el comportamiento de los circuitos es imprevisible.

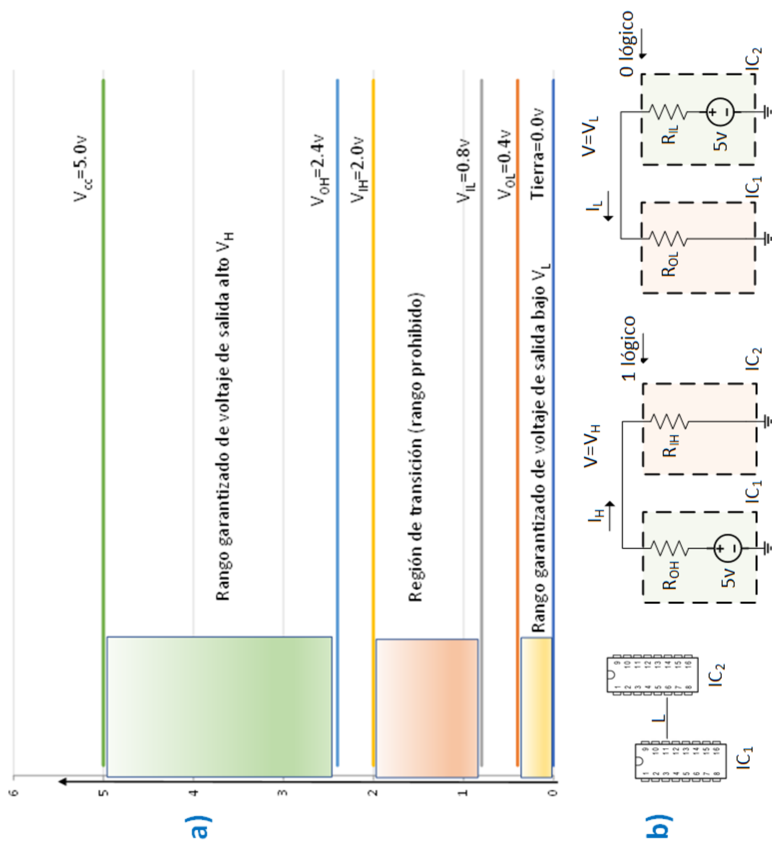


Figura 6.66 Circuito equivalente de voltaje entrada/salida.

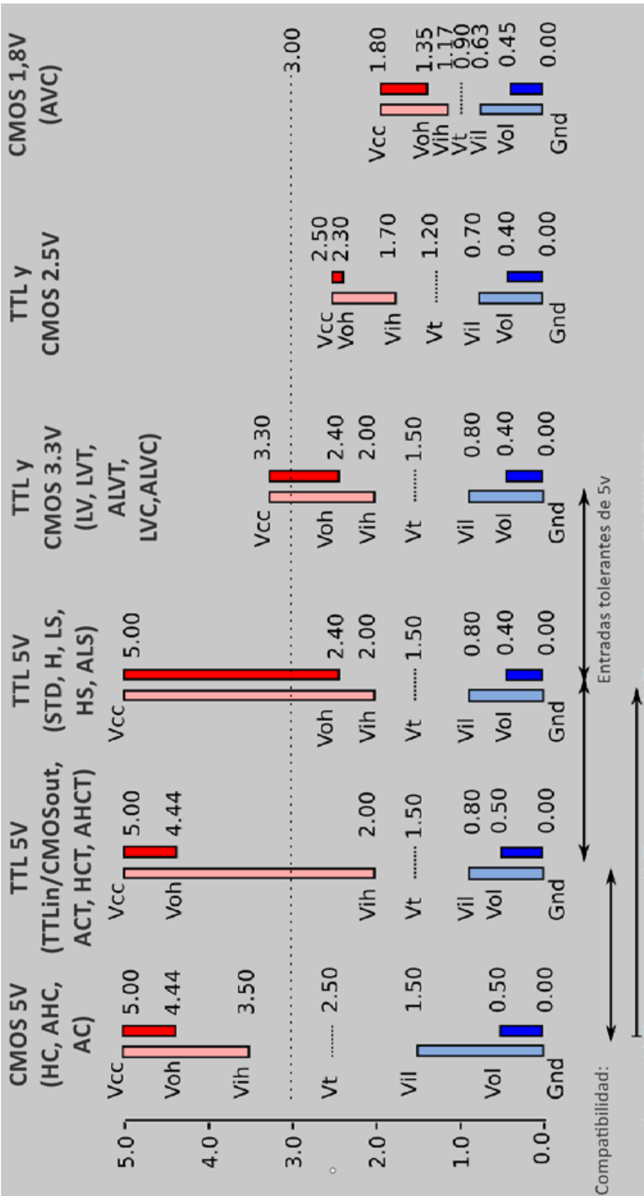


Figura 6.67 Voltajes de las distintas familias.

Las señales de salida producidas por un circuito estándar ***TLL*** son usualmente garantizados para caer en una región más estrecha que se muestra sombreada en la figura *6.7a*. En la práctica, los voltajes fluctúan alrededor de un valor preestablecido por diferencias en la alimentación de voltaje o por pequeñas variaciones en las características de los componentes e interacciones imprevisibles entre distintas señales; todos estos efectos son llamados de forma colectiva ***ruido***. La diferencia de 0.4V entre el valor en el

peor de los casos de la salida garantizada y el valor aceptable de diseño (de 2.0 a 2.4V y de 0.4 a 0.8V) es llamada ***margen de ruido***.

Suponga que la salida de un circuito IC_1 es conectada a la entrada de otro IC_2 tal como se muestra en la figura *6.7b*. Se dice del circuito IC_1 que ***maneja*** al circuito IC_2, mientras que IC_2 ***carga*** a IC_1. IC_1 transmite la información lógica a IC_2 causando que el voltaje aplicado por medio de la línea *L* asuma valores entre los rangos V_H y V_L. Correspondiente a estos niveles de voltajes, fluyen corrientes eléctricas denotadas como I_H e I_L. Los valores exactos que estos voltajes y corrientes tienen son determinados por las características eléctricas de los circuitos, en particular por la resistencia ofrecida a las señales de entrada y salida. El comportamiento electrónico de los circuitos puede ser analizado por medio de los circuitos equivalentes representados en la figura *6.7b*, que muestra versiones simplificadas de las etapas de salida y entrada encontrados en las varias familias descritas anteriormente.

En la segunda parte de la figura *6.7b* se muestra la situación en la que el circuito IC_1 transmite un uno lógico poniendo el voltaje de la línea en V_H. Una corriente I_H llamada corriente fuente, fluye de un circuito a otro. De forma similar cuando (ver figura *6.7b* tercera parte) un cero lógico es transmitido del circuito IC_1 al IC_2, una corriente I_L, llamada corriente de sumidero, fluye hacia IC_1. Nótese que la dirección del flujo de la señal lógica y el flujo de la corriente son independientes y no deben confundirse. El valor del voltaje *V* que aparece en la línea se determina por las siguientes ecuaciones que son obtenidas directamente de los circuitos equivalentes de la figura *6.7b*.

$$V_H = I_H R_{IH} = 5 - \frac{5R_{OH}}{R_{IH}+R_{OH}} \quad (6.1)$$

$$V_L = 5 - I_L R_{IL} = \frac{5R_{OL}}{R_{IL}+R_{OL}} \quad (6.2)$$

De estas ecuaciones podemos deducir que, conforme la carga resistiva de entrada $\frac{R_{IH}}{R_{OL}}$ impuesta por el circuito integrado C_1 decrece, V_H decrece y V_L crece, mientras las corrientes I_H e I_L decrecen ambas. Si la resistencia de entrada del primer circuito es demasiado pequeña, implicando que los circuitos son incompatibles, los valores no transitivos del voltaje *V* en la línea pueden moverse hacia la región prohibida, resultando así, en un comportamiento lógico indeterminado. La incompatibilidad puede resultar también en valores excesivos de la corriente que pueden dañar a cualquiera de los circuitos.

Para prevenir tales incompatibilidades los fabricantes de circuitos integrados especifican valores mínimos y máximos permitidos para los voltajes y corrientes aplicados a cada una de las líneas de entrada y salida de cada circuito integrado. Estos valores límites son los mismos para todos los miembros de una familia lógica.

Mucha información adicional que es útil para el diseñador de sistemas puede encontrarse en las hojas de especificación de los fabricantes. Por ejemplo, en la familia lógica ***TLL*** el voltaje de la fuente de poder puede variar solamente en un 5% (los circuitos militares permiten una variación del 10%) del valor nominal de 5 voltios. Las especificaciones nos muestran también el máximo consumo, así como el máximo número de compuertas que pueden ser conectadas a la salida o a la entrada.

Las restricciones en la capacidad del manejo de salida o de entrada (fan−out, fan−in) se pueden analizar por medio de los circuitos equivalentes de la figura *6.7*. Por simplicidad supongamos que los dispositivos IC_0, IC_1, ..., IC_k corresponden todos a la misma familia lógica. Cada uno de los dispositivos de carga pueden tomar una corriente máxima de I_{IH} de la línea, por lo que la corriente de carga, en el peor de los casos, tomada del dispositivo 0 es $\mathrm{k}I_{\mathrm{IH}}$. La operación apropiada del circuito se garantiza si se cumple la desigualdad:

$$I_{OH} \geq kI_{IH} \qquad (6.3)$$

De la misma forma encontramos la segunda desigualdad:

$$I_{OL} \geq kI_{IL} \qquad (6.4)$$

Estas desigualdades se pueden generalizar fácilmente para cubrir distintas familias lógicas.

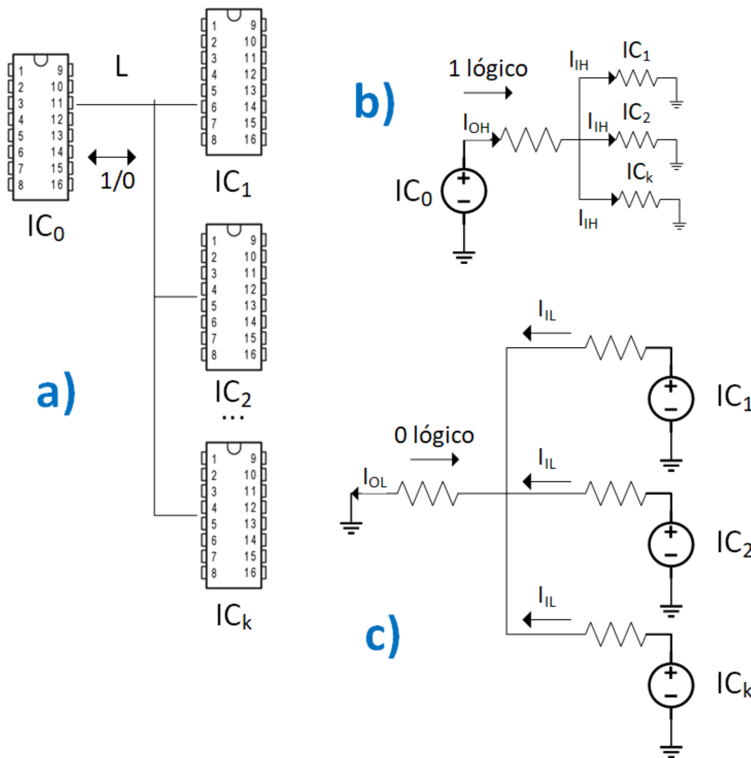


Figura 6.68 Circuitos equivalentes de manejo salida/entrada.

Suponga que los circuitos de la figura *6.9* son ***CI*** (circuitos integrados) de la serie 7400 de la familia ***TTL***. Si buscamos en las hojas de especificaciones los valores correspondientes de corriente y voltaje los podemos sustituir en las ecuaciones anteriores para encontrar una capacidad de manejo de salida de 10 y una de entrada de 5, por lo que la segunda ecuación limita el número total a 5 compuertas.

En el diseño de sistemas digitales frecuentemente se encuentra uno donde una línea ***L*** tiene una corriente de salida máxima insuficiente para manejar todas las líneas de entrada que deben conectarse a ella. Un circuito que amplifique la corriente llamado ***amplificador*** (buffer) puede ser usado para rectificar este problema. El amplificador se representa con un triángulo en los diagramas lógicos y se inserta a la línea ***L*** para aumentar la corriente de salida disponible. Muchas veces las compuertas del tipo ***NO*** se usan como amplificadores.

6.5 Consideraciones de Ruido

Pocas materias son tan inclusivas en su ámbito y con tanta variedad en su connotación como lo es el generalizado término del

ruido. Si en un circuito hace su presencia el ruido, podemos llegar a extremos en los que, bajo ciertas condiciones, se produzcan respuestas erróneas – usualmente peor que si no hubiese respuesta. En cambio, existen otros tipos de ruidos que, aunque no deseados, no son realmente dañinos.

6.5.1 Tipos de Ruidos y Métodos de Control

En el diseño lógico digital se debe tratar con muchos tipos de ruidos. Las siguientes clasificaciones de ruido son de gran utilidad (lista no exhaustiva):

1. Ruido externo o de interferencia. Generado por los dispositivos electrónicos como consecuencia de su naturaleza física (ruido blanco, ruido térmico (Johnson–Nyquist), ruido cuántico, etc.). Es de naturaleza aleatoria.
2. Ruido de fluctuación. Aquel radiado por el sistema debido a su actividad inherente. Tal ruido puede ser causado entre otros por interruptores, motores de escobillas, contactos que se abren o cierran, señales que se transmiten de un lado a otro del sistema (ruido rosa).
3. Ruido de la Fuente de Alimentación. Ruido del acoplamiento de la alimentación de voltaje de ***CD*** o ***CA***. Sus fuentes son usualmente similares a las del ruido externo.
4. Ruido Cruzado (cross−talk). El que se induce en las líneas conducción de señales debido a la transmisión de información en otras líneas cercanas.
5. Ruido de la corriente de señal. Ruido generado por las impedancias parásitas que se forman en el circuito al circular la corriente.
6. Reflexiones de las líneas de transmisión. Ruido de líneas de transmisión que causan fallas intermitentes si no se acoplan correctamente sus impedancias y cargas.
7. Picos de corriente de la fuente de alimentación de un circuito. Causados en circuitos del tipo ***TLL*** por su salida de tótem, si su diseño no es correcto.

En general estos tipos de ruido se tratan por los siguientes métodos:

Tipo de Ruido	***Tratamiento***
Externo	Blindaje, Aterrizaje, Desacoplamiento.
Línea de alimentación	Blindaje, Aterrizaje, Desacoplamiento.
Ruido Cruzado	Blindaje, Aterrizaje, Desacoplamiento, Propiedades de los circuitos.
Corriente de la señal	Blindaje, Aterrizaje, Desacoplamiento, Propiedades de los circuitos.
Línea de transmisión	Propiedades de los circuitos, diseño.
Picos de corriente	Propiedades de los circuitos, diseño.

6.5.1.1 Blindaje

Un equipo eléctrico puede funcionar en un ambiente ruidoso además del propio ruido que se pueda generar internamente. Los pulsos de ruido proceden de distintas fuentes electrostáticas, electromagnéticas o ambas y su frente de onda debe mantenerse fuera del equipo. Una técnica utilizada para evitar que interfieran con los circuitos es rodear al circuito de un material metálico llamado blindaje. Usualmente los campos que generan el ruido cambian rápidamente por lo que el blindaje necesario para excluirlos es pequeño. Aunque se puede usar aluminio para este propósito, es mucho mejor y más práctico utilizar un material ferroso y conectar éste a una buena tierra externa por medio del cable de alimentación.

6.5.1.2 Aterrizaje y Desacoplamiento

Si la línea de un dispositivo que transmite a otro se encuentra cercana al circuito, no se da pie a una discontinuidad en la señal que se transmite; si, por el contrario, la línea no se regresa por medio de una buena conexión a tierra, el efecto es una discontinuidad que el circuito interpreta como una alta impedancia que genera ruidos.

Si el retorno a tierra se realiza adecuadamente, se obtienen buenos resultados y existe una cancelación de la corriente en el punto de tierra eliminando los picos y disparos no deseados de voltajes que pueden ocasionar ruidos.

Dos reglas disminuyen los efectos de una línea de transmisión a niveles aceptables en las familias lógicas:

1. Usar cable trenzado o coaxial[29] para las líneas de transmisión y aterrizarlas cerca del circuito transmisor y receptor.
2. ***Desacoplar*** la señal de voltaje por medio de un pequeño capacitor cerca de los circuitos transmisor y receptor.

La acción de utilizar capacitores pequeños que puedan compensar el cambio de corriente y absorber los transitorios de un estado a otro es una práctica común en el diseño lógico. Se recomienda un pequeño capacitor por cada circuito (0.01µF) o uno un poco más grande por cada grupo de circuitos.

El uso de una tierra común de retorno para todos los circuitos requiere de mucha atención y es usual utilizar los siguientes procedimientos:

1. Hacer la pista (en una tarjeta de circuito impreso) de tierra tan ancha como sea posible, aunque esto signifique que haya grandes cambios en su ancho.
2. Formar un lazo completo alrededor de la tarjeta y unir ambos planos, en tarjetas de dos lados, por medio de conectores separados que lleven a una tierra común.
3. Llevar los dos puntos anteriores a un extremo y utilizar un lado de la tarjeta para las conexiones y el otro (el lado de los componentes) para un plano de tierra completo que sólo se interrumpa en donde los componentes se insertan en la tarjeta para soldarse.

6.5.1.3 Reflexiones en Líneas de Transmisión

Cuando las interconexiones usadas para transmitir información digital llegan a ser muy largas, existe un efecto en el que el retraso en la propagación de la señal (por los efectos de los parámetros propios de la línea) es comparable con el del cambio de estados en la señal (frecuencia). Cuando esto sucede, se deben considerar efectos de reflexión donde parte de la onda que llega al lado transmisor "rebota" hacia el lado transmisor, creando interferencias entre las mismas señales.

---

[29] El cable coaxial, creado en la década de 1930, es un cable utilizado para transportar señales eléctricas de alta frecuencia que posee dos conductores concéntricos, uno central o núcleo, encargado de llevar la información, y otro externo de aspecto tubular, llamado malla, blindaje o trenza, que sirve como referencia de tierra y retorno de las corrientes. Entre ambos se encuentra una capa aislante dieléctrica, cuyas características dependerán de la calidad del cable. Todo el conjunto suele estar protegido por una cubierta aislante llamada camisa exterior.

Se han establecido algunas reglas para minimizar este efecto:

1. Usar interconexiones directas de alambre que no tengan una línea de retorno para distancias no mayores a 30cm.
2. Las conexiones directas de alambre que excedan 30 cm deben ir junto a un plano de tierra, pero no deben exceder 60cm.
3. Asegurarse de que existe un plano de tierra próximo a la línea de transmisión.
4. Desacoplar los circuitos de la fuente de voltaje por medio de capacitores pequeños (0.001µF).
5. Para líneas largas usar, cable coaxial de la impedancia requerida por los circuitos. Altas impedancias aumentan el ruido cruzado y bajas impedancias son difíciles de controlar y requieren de más potencia.

6.6 Resumen

Se realiza un estudio de las distintas familias lógicas y la forma que tiene cada una de ellas para realizar circuitos prácticos de las funciones lógicas más comunes.

Se analiza las diferencias, ventajas y desventajas de cada familia con respecto a las otras y se dan las características en detalle de dos familias muy en uso hoy en día como son la familia ***TTL*** y la ***MOS*** (***PMOS*** y ***CMOS***).

Las características eléctricas de las familias lógicas son analizadas en cierto detalle, así como su interconexión y los distintos parámetros utilizados en lógica digital.

Se introducen símbolos estándar para marcar los voltajes y corrientes existentes en los circuitos lógicos.

Se consideran los orígenes y aspectos del ruido, así como las características pertinentes de los dispositivos utilizados para eliminarlos o contraatacarlos en los sistemas lógicos.

6.6.1 Puntos Importantes del Capítulo

- Los circuitos electrónicos digitales se dividen en familias.
- Las familias en uso más extendido hoy en día son la ***TTL*** y ***CMOS***.

- La familia ***TTL*** tiene alta velocidad, pero también alto consumo de corriente; la familia ***CMOS*** tiene bajo consumo de corriente, pero es lenta.
- Una de las técnicas para mejorar la velocidad de la familia ***CMOS*** es bajar su voltaje.
- La familia ***TTL*** maneja niveles de voltaje entre 5 y 0 Voltios para indicar el 1 y el cero lógico.
- A la capacidad de “manejar” x número de compuertas se le denomina manejo de salida (fan−out). A la capacidad de ceder corriente a otros circuitos cuando éstos están en 0 lógico se denomina manejo de entrada (fan−in).
- Se puede usar un amplificador para poder manejar más carga a la salida.
- A las interacciones imprevisibles entre señales y a las fluctuaciones de un voltaje de valor preestablecido se les conoce como ruido.
- El ruido no se puede evitar, pero es posible controlarlo y combatirlo con algunas técnicas como desacoplamiento, blindaje, aterrizaje, etc.

7 Elementos Lógicos El Flip–Flop

Presentaremos a continuación una variedad de dispositivos electrónicos utilizados en la construcción de circuitos lógicos y computadoras digitales. El análisis que se da de cada componente es breve y para una mayor profundidad sugerimos consultar la bibliografía que aparece al final del libro.

7.1 Flip–flop Tipo SR

El circuito de la figura *7.1a* presenta un par de compuertas del tipo ***NOO*** en una configuración conocida como ***flip–flop*** o ***biestable***. Tiene un par de terminales llamadas ***S*** y ***R***, que corresponden a las palabras en inglés *Set* (activa) y *Reset* (inactiva) respectivamente. Usaremos los símbolos ***S*** y ***R*** no sólo para indicar las terminales sino también para especificar su estado lógico. Así, si ***S=1*** indica que el voltaje correspondiente a un nivel lógico de 1 está presente en la terminal ***S***. De forma similar las salidas ***Q*** y $\bar{Q}$ indican las terminales y su estado lógico. En esta notación hemos incluido el hecho de que las salidas son complementarias en la operación normal del circuito.

En la figura *7.1* encontramos representados:

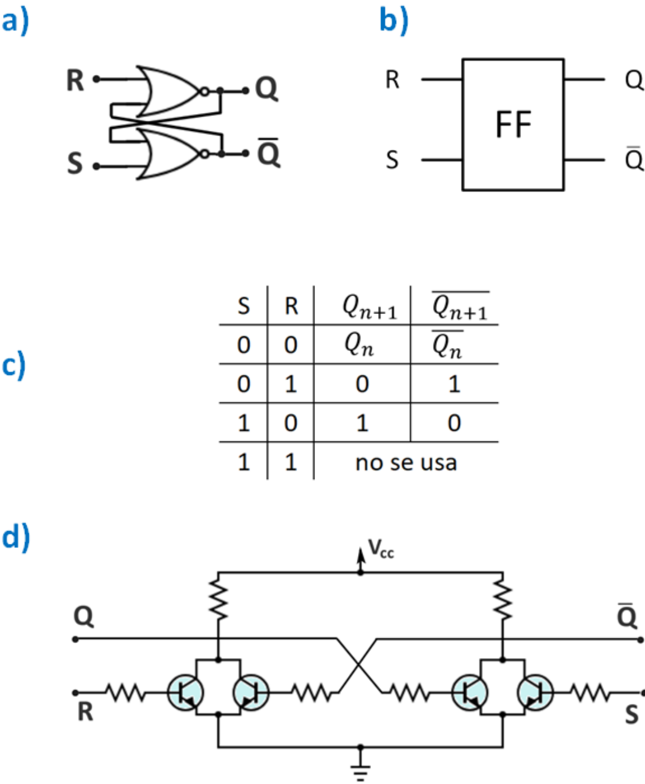
a) Un flip-flop tipo *SR* implementado con compuertas tipo ***NOO***. Tenga en cuenta que este mismo circuito puede realizarse con compuertas tipo ***NOY*** con una ligera modificación a su entrada.
b) La representación típica de un biestable *SR*.
c) La tabla de verdad del biestable tipo *SR*.
d) La realización de un biestable SR usando lógica ***RTL*** (resistencia-transistor).

Flip-Flop

El primer flip-flop electrónico fue inventado en 1918 por los físicos Ingleses William Eccles y F. W. Jordan. Se llamaba *Circuito de Disparo Eccles–Jordan* y consistía en dos tubos de vacío. Fue utilizado en la computadora rompe códigos *Colossus* fabricada en 1943.

Biestable

Un biestable (en inglés Flip-Flop, si es síncrono, y latch, si es asíncrono), es un circuito capaz de permanecer en uno de dos estados posibles durante un tiempo indefinido en ausencia de perturbaciones. Esta característica es ampliamente utilizada en electrónica digital para memorizar información. El paso de un estado a otro se realiza variando sus entradas.



S	R	Q_{n+1}	$\overline{Q_{n+1}}$
0	0	Q_n	$\overline{Q_n}$
0	1	0	1
1	0	1	0
1	1	no se usa	

Figura 7.69 Flip-Flop SR con compuertas NOO.

La característica fundamental y más importante de un biestable es que tiene "memoria". Esto es, dado que los estados en las entradas ***S*** y ***R*** son 0 en cierto momento, es posible, examinando la salida, saber el estado en un tiempo anterior de las entradas inmediatamente antes de llegar a su nivel presente.

7.1.1 Terminología

Para comprender mejor la descripción que a continuación hacemos de distintos circuitos flip−flop, es conveniente conocer la actitud que prevalece entre diseñadores de sistemas lógicos.

En las compuertas del tipo ***NOY*** y ***NOO*** (como en las ***Y*** y las ***O***), cuando sirva a nuestros propósitos, podemos designar arbitrariamente a una de las terminales de entrada como una entrada de habilitación−deshabilitación (enable−disable input). Por lo que, si consideramos a una compuerta ***NOO*** u ***O***, si una de las entradas seleccionadas tiene un 1 lógico, la salida de la compuerta es independiente de las otras entradas. Esta sola entrada toma control de la compuerta y la compuerta está deshabilitada con respecto a

las otras entradas (el término inhibir se usa en el mismo sentido que deshabilitar). Alternativamente, si la entrada es un 0 lógico, la entrada escogida no toma control de la compuerta y ésta está habilitada para responder a sus otras entradas. En una compuerta del tipo ***NOY*** o ***Y***, una entrada seleccionada toma control y deshabilita a las demás cuando su valor lógico es 0, puesto que con una sola entrada que tome el valor de 0 lógico, la compuerta no puede responder a otras entradas y su salida será siempre 0. Resaltamos que en el primer caso (compuerta ***NOO*** y ***O***) la entrada toma el control cuando tiene el valor lógico de 1 y en el caso de las compuertas del tipo ***NOY*** e ***Y*** cuando tiene el valor de 0 lógico.

Es también común ver al voltaje que designa al 0 lógico como un estado básico, sin disturbios, no perturbado, en el que “nada ha pasado” y al voltaje que designa el 1 lógico como el estado excitado, activo, efectivo, etc., al que se llega después de que “algo ha pasado”.

7.1.2 Funcionamiento

En el circuito del flip−flop, las entradas ***S*** y ***R*** son las de control. Si estamos tratando con un circuito formado por compuertas del tipo ***NOO*** las entradas ejercen control sólo si están en el estado lógico correspondiente a un 1. Si suponemos que *S=R=0* podemos ignorar esas entradas y concentrarnos en la salida del circuito que forman las otras dos entradas de las que sí dependen las compuertas. Si suponemos que $\bar{Q} = 1$, la salida de la compuerta superior es un cero que es consistente con los símbolos utilizados y *Q=0*. Supusimos en un principio, para llegar a este resultado, que $\bar{Q} = 1$, y ahora debemos comprobar esta suposición. Observemos que las dos entradas a la compuerta inferior son 0 por lo que su salida es 1.

Si suponemos ahora que $\bar{Q} = 0$, y que *S=R=0* llegaremos a que *Q=1* consistente nuevamente con nuestra notación. De lo que concluimos que si *S=R=0* el biestable persiste en cualquiera de las dos situaciones (llamadas usualmente estados estables), en uno con $Q = 0$; $\bar{Q} = 1$ y en otro $Q = 1$; $\bar{Q} = 0$. Llamaremos al primero como estado “apagado” (reset o clear) mientras al otro “prendido” (set).

Los otros dos estados (de la tabla de la figura *7.1c*) pueden ser verificados de la misma forma. Se dice que el flip−flop tiene memoria porque si en un tiempo ***t*** las entradas cambian a prendido

o a apagado, y en el tiempo *t+1* las entradas vuelven a su estado estable (de 0 o apagados), analizando las salidas ***Q*** y $\overline{\boldsymbol{Q}}$ podemos deducir qué entrada se necesitó para que llegasen a su estado presente. En otras palabras, con las entradas del flip−flop *S=R=0*, el estado del dispositivo depende del estado anterior inmediato de las entradas.

El estado, cuando las entradas *S=R=1*, no es utilizado pues las salidas no corresponden una al complemento del otro y si las entradas cambiasen a *S=R=0* el estado final de las salidas no se podría predecir.

7.1.2 Flip−Flop con Compuertas del Tipo NOY

En la figura *7.2* mostramos el mismo flip−flop realizado con compuertas del tipo ***NOY***. Puesto que se usan compuertas ***NOY***, las entradas ejercen el control cuando se encuentran a nivel lógico de 0, esto es, el flip−flop cambia de estado cuando cualquiera de las entradas cambia a 0 lógico. Debido a esto, se han etiquetado las entradas como $\bar{S}$ y $\bar{R}$ en lugar de ***S*** y ***R***. Debido a incompatibilidad con la tabla anterior del flip−flop, podemos imaginar las entradas como dos variables de los que $\bar{S}$ y $\bar{R}$ son complementos para llegar a la tabla genérica del flip−flop (figura *7.1c*) que se aplica independientemente de las compuertas que intervengan en la realización del circuito.

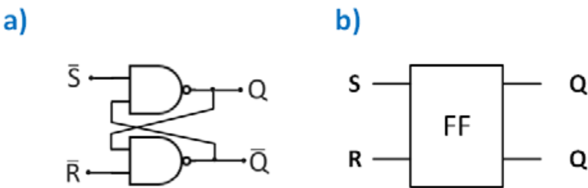


Figura 7.70 Biestable SR con compuertas NOY.

c)

$\bar{S}$	$\bar{R}$	Q_{n+1}
0	0	no se usa
0	1	0
1	0	1
1	1	Q_n

7.1.3 Aplicación

Una de las aplicaciones sencillas, pero prácticas de los flip–flops es la que se muestra en la figura *7.3*. En ella encontrará el diagrama de un interruptor que cambia de una posición a otra (figura *7.3a*). El cambio de posición del interruptor no se realiza instantáneamente y los componentes del pulsador, aunque no se aprecie a simple vista, tienen un rebote mecánico en el que la corriente atraviesa las placas del interruptor varias veces. Esto sucede muchas veces hasta que el efecto mecánico se atenúa y los contactos del interruptor quedan quietos en su posición final fija. Este fenómeno sucede tanto al cerrar como al abrir el interruptor.

En un circuito digital, cada uno de estos "choques" entre los contactos indica al circuito que le sigue que tome alguna acción. La señal resultante no es un cambio único de 1 a 0 (o viceversa), como pudiese esperarse, sino una cadena de unos y ceros. Tantos como el circuito siguiente detecte la señal precedente como un cero o uno válido si el tiempo de respuesta es el adecuado.

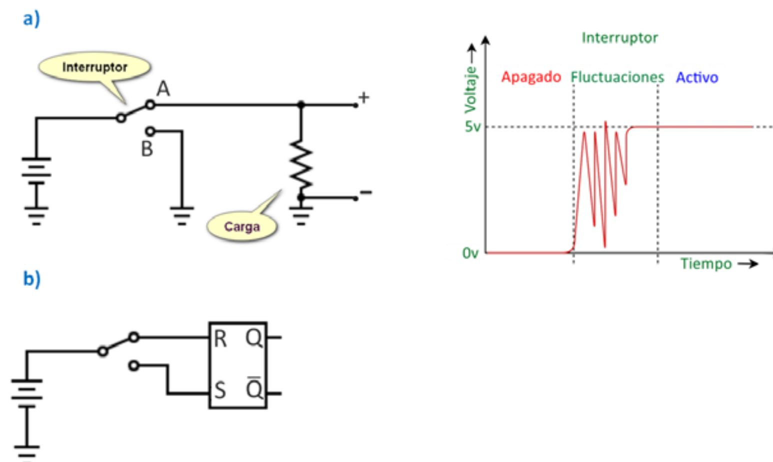


Figura 7.71 Aplicación común de un biestable.

Es una práctica común el utilizar un flip–flop en el circuito siempre que intervengan interruptores mecánicos para evitar estos "rebotes" y tener una señal limpia a la salida del circuito. A este tipo de circuito se le denomina interruptor sin rebotes (chaterless switch).

7.1.3.1 Otras Formas de Atenuar el Rebote

Aunque el circuito anterior resuelve muchos de los problemas del "rebote" en los interruptores (considere el teclado de una

Dependiendo del tipo de entradas los biestables se dividen en:

- Asíncronos: solamente tienen entradas de control. El más empleado es el SR.
- Síncronos: además de las entradas de control posee una entrada de sincronía o de reloj.

Si las entradas de control dependen de la de sincronismo se denominan síncronas y en caso contrario asíncronas. Por lo general, las entradas de control asíncronas prevalecen sobre las síncronas.

La entrada de sincronismo puede ser activada por nivel (alto o bajo) o por flanco (de subida o de bajada). Dentro de los biestables síncronos activados por nivel están los tipos SR y D, y dentro de los activos por flancos los JK, T y D.

Los biestables síncronos activos por flanco se crearon para eliminar las deficiencias de los latches (biestables asíncronos o sincronizados por nivel).

computadora con 100 o más interruptores), se considera una solución cara en ciertos casos y se prefieren otras:

- Un simple circuito RC puede, en algunos casos, convenir para aliviar el problema y resolverlo por completo. Bastará para ello escoger cuidadosamente el tiempo de carga y descarga del circuito (constante R*C) para que su tiempo sea similar al del establecimiento del contacto (el que demora la lámina móvil en detenerse) del interruptor.
- Si las dos soluciones propuestas hasta el momento no logran convencer al diseñador, existe una tercera, por medio de programación, que consiste en incluir una técnica llamada de ***cambio de estado demorado*** que cambia el estado de la entrada conectada al interruptor un cierto tiempo después de que se haya detectado el primer cambio, ignorando todo cambio intermedio que exista.
- Dentro de las soluciones programáticas existe también la llamada ***cambio de estado confirmado*** que consiste en cambiar de estado solamente luego que un cierto número de muestras periódicas y consecutivas de la entrada coincidan en un nuevo valor (0 o 1). El número de muestras se debe elegir en función de la frecuencia de muestreo y del tiempo de establecimiento del interruptor.

7.2 El Reloj

Los circuitos con reloj, síncronos o de paso a paso, forman la mayoría de los diseños lógicos digitales. Una señal que sincroniza a todas las demás es muy conveniente para organizar todos los eventos que suceden y asegurarse de que nada cambia hasta que se dé la señal correspondiente. La razón para llamar a esta señal reloj es debido a su rápida variación y a que simula el tic tac de un reloj de pulso marcando el compás de cada operación.

En lugar de cambiar el estado de un circuito en cuanto se tenga una señal a la entrada de él, se prefiere el arribo de un pulso del reloj; es entonces que el circuito responde a las entradas y nos da una salida confiable.

Existen varias ventajas al contar con una lógica dependiente del reloj:

- Las condiciones no verificadas no pueden propagarse por todos los demás circuitos. Es lugar de esto, los cambios

suceden ordenadamente, una etapa a la vez. Estas acciones, de un paso a la vez, hacen posibles circuitos tales como contadores y registros de corrimiento (analizados en el capítulo *8*).

- Todo cambio en el sistema ocurre más o menos al mismo tiempo (hay que considerar las características de retraso de cada circuito en particular), esto es, de forma síncrona. Esto elimina o minimiza las condiciones llamadas de carrera (donde la salida de un circuito va corriéndose sin control por los demás), falsos disparos (glitches) y confusión en las secuencias de tiempo.
- Las señales de reloj se generan con circuitos especiales o usando un cristal de cuarzo que oscila en una frecuencia muy alta y exacta que luego se divide. Analizaremos brevemente los circuitos de reloj en el capítulo 9.

La tasa de cambio del reloj (cuántas veces varía por segundo) es conocida como frecuencia del reloj y se mide en Hertz:

(7.1) $$f = {}^{1}/_{t} \ (Hz)$$

donde ***f***=frecuencia y ***T***=periodo.

A cada uno de los cambios del reloj se le conoce como pulsos y en muchos sistemas estos pulsos se generan de forma regular e ininterrumpida lo que hace aún más claro la denominación de reloj a este tipo de señales.

7.3 Flip−flop SR con Reloj

Analicemos la situación mostrada en la figura *7.4a* para entender el problema común que lleva al uso de los relojes como mecanismos de sincronización entre actividades que se suceden en los circuitos. Se asume que las entradas al flip−flop se encuentran en 0 y ***S*** es el resultado combinacional de ***A*** y ***B*** cuya salida del flip−flop es *Q=0*.

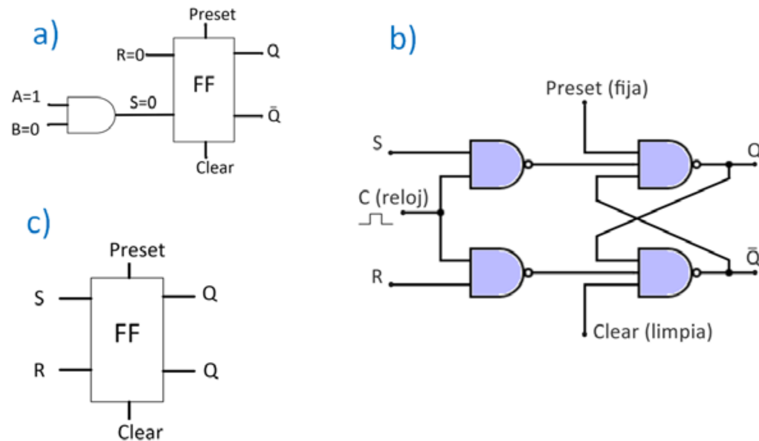


Figura 7.72 Biestable con reloj, limpia y fija.

Supóngase que las entradas a la compuerta ***Y***, ***A*** y ***B***, cambian casi al mismo tiempo y que no es nuestra intención que la salida final de la compuerta cambie el estado en el que se encontraba el flip–flop al iniciar la secuencia. Si ***A*** cambia antes que ***B*** o viceversa, encontramos que la salida de la compuerta, que es la entrada del flip–flop, cambia su estado. Es sumamente difícil poder diseñar un circuito que calcule todos los retardos de las señales, de forma que se pueda asegurar una simultaneidad en los resultados de eventos que se planeen en el desarrollo de un circuito lógico digital. En sistemas grandes, el retraso de las señales a lo largo del camino que deben recorrer causa dificultades del tipo descrito anteriormente, esto es que *A* cambie ligeramente antes que *B* o viceversa.

El retraso causa dificultades aún en sistemas combinacionales, pero es más notorio en circuitos que constan de algún tipo de memoria. Para evitar estos contratiempos de diseño se escoge utilizar una señal adicional que le diga a un circuito que todo está estable y que se puede proceder a analizar las entradas para ejecutar la función para la que el circuito fue diseñado. Las transiciones se difieren hasta que todo el sistema alcanza un estado estable en el que no tengamos sorpresas.

Un flip–flop que integra este tipo de control se muestra en la figura *7.4b*. Se han agregado dos compuertas al circuito básico que nos permiten tener control sólo cuando la entrada ***C*** (reloj, Clock) cambia de 0 a 1. Como el circuito únicamente responde en la transición de la entrada ***C*** cuando ésta cambia de 0 hacia 1, se le denomina un circuito *Síncrono* (porque ahora depende de una señal auxiliar) que responde al borde positivo del reloj. Las entradas

auxiliares ***LIMPIA*** (clear) y ***FIJA*** (preset) nos sirven para poner al flip–flop en un estado inicial predeterminado y son asíncronas, esto es, no dependen del reloj.

Se prefiere usar el circuito mostrado en la figura *7.5* pues en el de la figura *7.4* las entradas ***LIMPIA*** y ***FIJA*** no son completamente independientes del reloj.

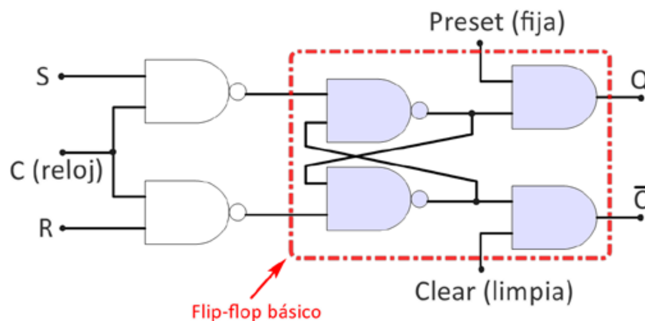


Figura 7.73 Flip-Flop SR alternativo con reloj, limpia y fija.

7.4 Flip–flop SR Maestro–Esclavo

Hemos visto que el circuito dependiente del reloj cambia su estado con la señal de sincronización, cuando ésta cambia de estable a activa (de 0 a 1). A esta permutación se le conoce como cambio del borde positivo del reloj. Existen muchas situaciones en que esta situación no es aceptable y se prefiere que el cambio se realice con el borde negativo del reloj, esto es, cuando el reloj cambia de activo a estable (de 1 a 0).

Las razones de esta preferencia obedecen a dos características muy generalizadas en los sistemas digitales:

1. Se usa un reloj común que sincroniza a todos los circuitos del sistema.
2. Los datos para alimentar a un flip–flop puede ser que se deriven de la salida de otros biestables.

El problema puede presentarse porque, además de que el biestable responde a las entradas presentes antes de la transición del reloj, responde también a las que resulten de los cambios sucedidos en otros circuitos durante la parte activa del reloj.

Para ahondar en este punto observe el sencillo ejemplo de la figura *7.6a*. Se consideran dos biestables que están conectados en serie donde la salida de uno es la entrada del siguiente. Los biestables mostrados son del tipo *SR* vistos hasta ahora donde la salida

responde al borde positivo del reloj, esto es, cambian sus salidas cuando el reloj va de 0 a 1. Supongamos que hemos puesto (por medio de dos entradas adicionales no mostradas en la figura: ***FIJA*** (PRESET) y ***LIMPIA*** (CLEAR)) a los biestables en un estado inicial donde $Q_1=0$ y $Q_2=0$.

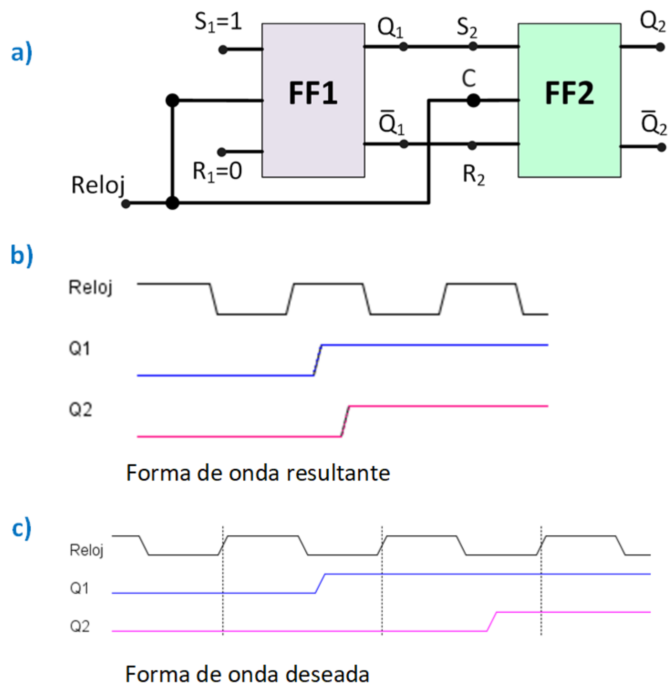


Figura 7.74 Efecto de carrera en un Flip-Flop SR.

Queremos que el circuito funcione de la siguiente forma: durante el primer ciclo del reloj, $\boldsymbol{Q_1}$ debe cambiar a 1 mientras la salida del segundo circuito debe permanecer inalterada. Durante el segundo ciclo del reloj, $\boldsymbol{Q_1}$ debe continuar valiendo 1 pero $\boldsymbol{Q_2}$ debe ahora cambiar a 1. Esto es, la entrada del primer circuito debe modificar el estado del primer circuito durante la primera fase del reloj y posteriormente, al segundo en la segunda fase del reloj.

Nos podemos ayudar para esto de una serie de diagramas (ver figura *7.6b* y *7.6c*) llamados ***diagramas de tiempos*** o ***cronogramas***, que nos muestran paso a paso las entradas y salidas de los circuitos. Observando la figura *7.6b* notamos que el resultado no es el descrito arriba (figura *7.6c*) sino que las salidas del segundo circuito responden durante la primera fase del reloj. Se han mostrado las ondas con un tiempo finito de transición de 0 a 1 y de 1 a 0 (como sucede en la realidad). Si existiesen otros circuitos en cascada conectados a la salida del segundo, sucedería el mismo fenómeno con ellos, pero llegaría un punto en el que a los

siguientes circuitos no les diera tiempo de responder y esto podría ser fatal para nuestro diseño.

Una de las formas de evitar esto sería hacer tan delgado el pulso del reloj que sólo permitiese que el primer circuito respondiera. La duración de los pulsos debería ser menor que el tiempo de retraso en cada uno de los circuitos.

Las desventajas de esta técnica son:

- Los pulsos de reloj pueden llegar a ser tan cortos que el primer flip−flop no responda de forma confiable.
- En circuitos muy rápidos es un problema generar pulsos de reloj de corta duración (delgados) de forma confiable y con la precisión requerida.

Una mejor solución resulta ser el uso de un nuevo circuito que responda a la caída o bajada del reloj en lugar de a su borde positivo. A estos circuitos se les denomina de borde negativo del reloj. Si nos referimos nuevamente a la figura *7.6a*, observamos que, si los circuitos responden a la caída del reloj, el segundo circuito no puede cambiar hasta el segundo pulso del reloj, que es como se estableció en un principio que nuestro diseño debe funcionar. Una segunda conveniencia es encontrada también cuando la salida del flip−flop es retroalimentada a sus entradas para obtener circuitos especiales que trataremos más adelante.

En el circuito de la figura *7.7a* se muestra un circuito que responde al borde negativo del reloj y al que se le denomina biestables SR de borde negativo o ***flip−flop maestro−esclavo*** y se forma de dos biestables normales que se conectan uno a la salida del otro y en el que el reloj aplicado al primero de ellos es negado en el segundo por lo que, cuando uno de ellos está inactivo, el otro se encuentra en su estado activo.

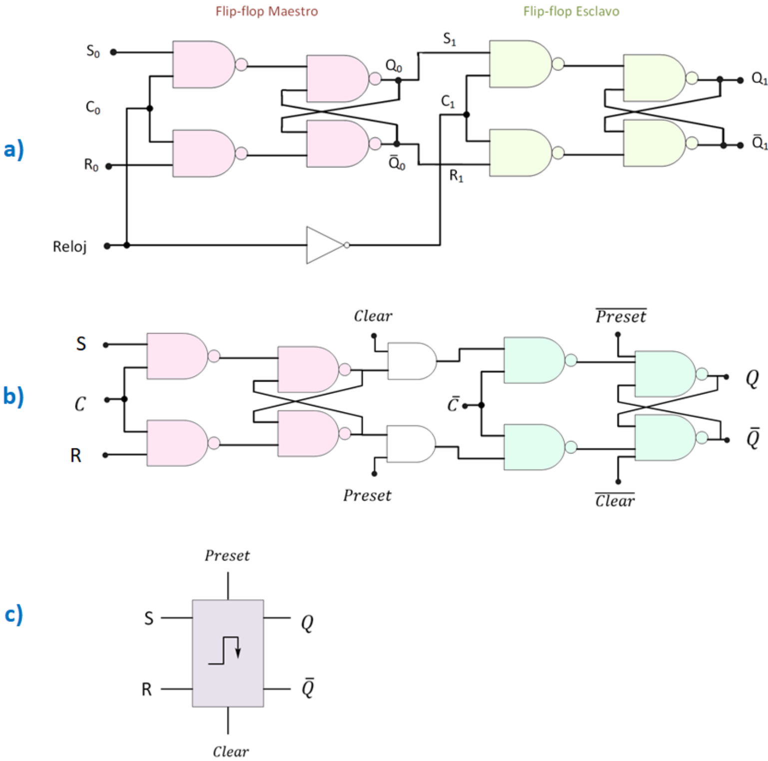


Figura 7.75 Flip-Flop SR maestro-esclavo.

La tabla de verdad de este nuevo flip−flop es idéntica a las vistas hasta ahora con la salvedad de que, en un circuito maestro−esclavo, la salida se encuentra disponible cuando el reloj realiza su transición de 1 a 0.

En el circuito de la figura *7.7b* mostramos el mismo flip−flop maestro−esclavo, pero con la ventaja de dos entradas adicionales que nos permiten iniciar la operación del biestable en un estado conocido. A estas dos entradas las denominamos ***LIMPIA*** (clear) y ***FIJA*** (preset) por la acción que realizan.

7.5 Flip−flop Tipo JK

En la figura *7.8a* mostramos un biestable con una pequeña modificación; la salida del flip−flop SR se retroalimenta a sus entradas por medio de unas compuertas ***NOY*** (pueden ser de otro tipo). Esta modificación cambia el funcionamiento del circuito y a las entradas se les nombra ***J*** (en lugar de ***S***) y ***K*** (en lugar de ***R***).

El objeto de la modificación es hacer que el circuito responda no sólo a las entradas y al reloj, sino a su salida también. Recuerde al analizar el circuito que éste no responde hasta que el reloj baja de

1 a cero, por lo que sus salidas son estables durante el pulso del reloj.

Otro aspecto importante de esta modificación es que el estado ***J=K=1*** sí puede ser utilizado y especifica que $Q_{n+1} = \overline{Q_n}$.

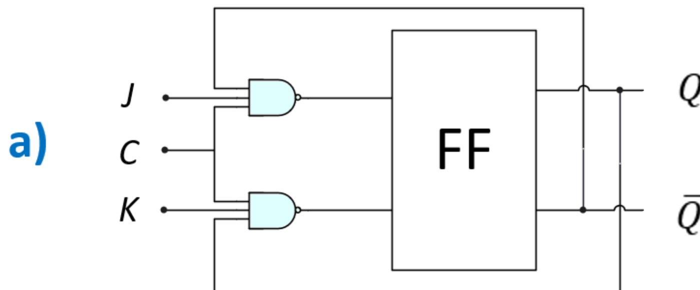


Figura 7.76 Flip-Flop tipo JK.

b)

J	K	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	$\overline{Q_n}$

7.6 Flip–flop Tipo D

Es frecuente que necesitemos retrasar una señal por la duración de un pulso de reloj; de ahí el nombre de un nuevo circuito: ***flip–flop Delay*** (retraso). La operación puede realizarse utilizando un único flip–flop ***SR*** o ***JK*** modificado ligeramente. El dato de entrada se aplica en forma directa a la entrada ***S*** o ***J*** y de forma complementada a ***R*** o ***K*** como se muestra en la figura *7.9a*. La forma de onda resultante (diagrama de tiempos) se muestra en la figura *7.9c*.

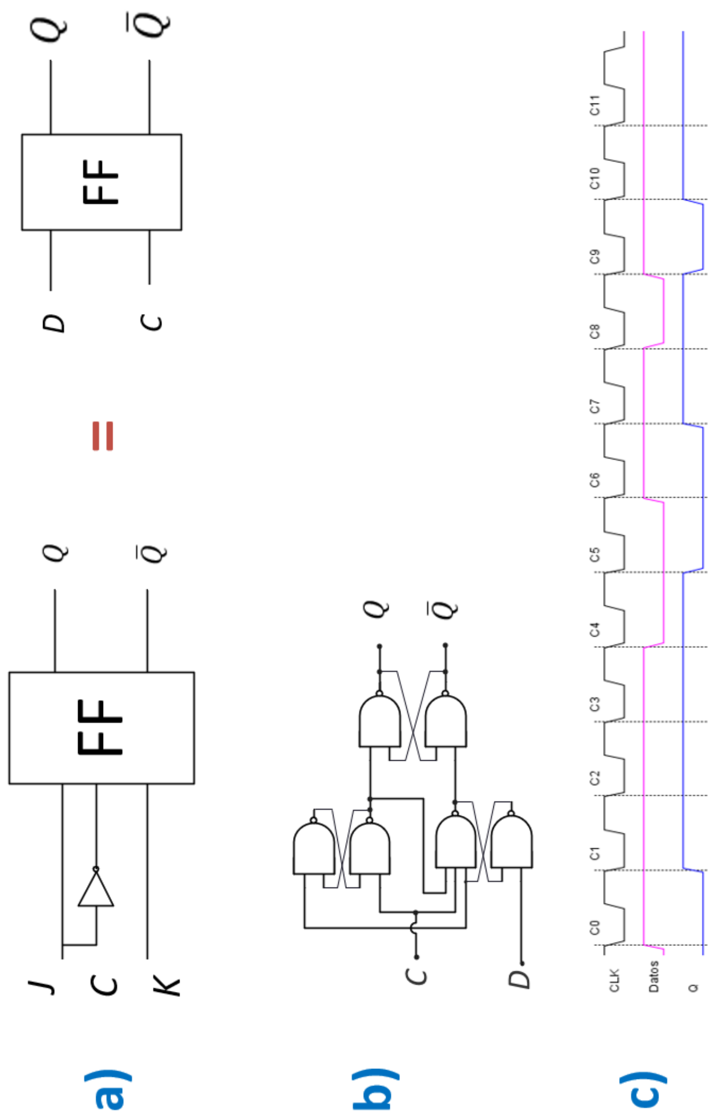


Figura 7.77 Flip-Flop tipo D.

Nuevamente se asume que el circuito responde al borde negativo del reloj. Se puede verificar de forma fácil que, dada una secuencia de entrada *x*, la salida ***Q*** es idéntica a ésta, pero retrasada un ciclo de reloj.

En el diagrama de tiempos se muestran las ondas como una transición exacta. En la práctica, los cambios tienen un tiempo corto pero finito que no coincide exactamente con la caída del reloj debido a retrasos y el tiempo que le lleva a los transistores cambiar de un estado a otro.

En la figura *7.9b* mostramos una realización con compuertas del tipo ***NOY***. El circuito responde con el borde positivo del reloj, pero las salidas inmediatamente deshabilitan cualquier otro cambio a las entradas, por lo que el efecto neto es similar a responder a la caída del borde negativo del reloj.

7.7 Flip-flop Tipo T

Existe un último tipo de flip-flop de gran utilidad llamado biestable tipo ***T***. El flip-flop ***T*** o conmutador (Toggle en inglés) cambia su salida con cada borde del reloj (cuando sube o baja según el diseño) haciendo que la salida sea igual a la mitad de la frecuencia de la entrada ***T***.

Su utilidad se manifiesta en la construcción de contadores binarios, divisores de frecuencia y, en general, dispositivos de suma binaria. Se puede fabricar a partir de un flip-flop tipo ***JK*** llevando ambas de sus entradas a un estado "alto" o 1 tal y como se muestra en la figura *7.10a*.

En la figura *7.10b* se muestra su diagrama de tiempos.

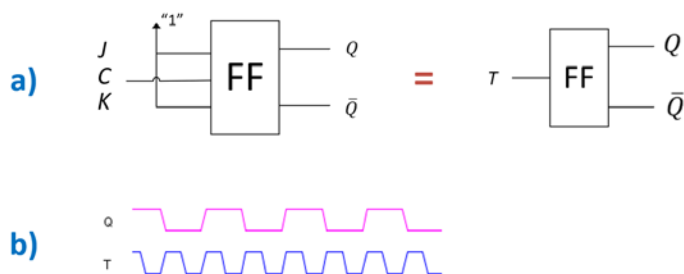


Figura 7.78 Flip-Flop tipo T.

Una de las aplicaciones para este tipo de flip-flops es en los contadores que trataremos en el siguiente capítulo.

7.8 ¿Dónde Usar los Flip–flops?

Muchos de los circuitos más complejos usan internamente grupos conectados de flip–flops tipo ***JK*** o ***D***, particularmente los contadores y registros de corrimiento (analizados en el siguiente capítulo). Hasta el momento, sólo se ha analizado el funcionamiento de los flip–flops pero sólo se han dado ejemplos concretos como parte de un interruptor sin ruido. Entre algunas de las aplicaciones en las que se usan biestables tenemos:

- *Memoria Provisional.* Se utiliza por lo general en despliegues numéricos y en toda circunstancia donde se deba guardar la información por un ciclo (flip−flops ***D***) o más de reloj (flip−flops ***JK***).
- *Eliminador de Ruido de Alta Velocidad*. Otra aplicación simple de almacenaje de los flip−flops es como parte de un circuito convertidor analógico a digital (convertidor A/D; analizados en el capítulo *9*). Aquí se usan como una memoria de almacenaje hasta que la conversión se ha realizado por completo y eliminado los ruidos de la conversión y transición.
- *Interruptor Sin Rebote*. Aplicación ya explicada en la sección *7.1.3*.
- *Muestra y Retiene (Sample and Hold)* . Un circuito de este tipo es uno analógico que mide el valor de entrada por un breve momento y guarda su valor para utilizarlo posteriormente. Si se trata de almacenar en forma analógica, la señal se pierde con el tiempo, de otra forma, digitalmente guardada en flip−flops puede durar indefinidamente con la exactitud que se desee. La señal de salida puede usarse como señal digital o convertirse de nuevo a analógica.
- *Eliminador de Basura*. Los circuitos de flip−flops como dispositivos de almacenamiento sencillos pueden usarse para remover porciones no deseadas o defectuosas de datos en sistemas de alta velocidad.
- *Secuenciadores y Registros de Corrimiento.* Analizados en el siguiente capítulo.
- *Divisores y Contadores*. Existen tantas posibilidades y son tan versátiles que se analizan en el siguiente capítulo.
- *Sincronizadores*. Los sincronizadores en general, son dispositivos que permiten alinear datos aleatorios del mundo externo del circuito con las secuencias internas regidas por un reloj propio del circuito.
- *Uno y sólo Uno*. Un circuito sincronizador que toma un evento externo sencillo; como podría ser el caso de un interruptor que se acciona o un pulso, en un pulso de precisión que dura un y sólo un intervalo entre periodos del reloj del sistema. Otra variación de este circuito es el de ***N y Sólo N***.
- *Resincronización*. Son circuitos que nos permiten eliminar retrasos de propagación y ruidos en compuertas y contadores. Permiten también convertir un evento no síncrono a uno en síncrono con el reloj del sistema.

- *Secuenciadores*. Reconocen una serie de eventos que deben suceder, por ejemplo, se pueden diseñar para reconocer la secuencia 10100 y ninguna otra.
- *Mezcladores*. El flip−flop tipo ***D*** puede usarse para producir a su salida la diferencia de dos frecuencias a sus entradas ***D*** y ***RELOJ***. La aplicación tiene sus limitaciones, pero aun así es una aplicación muy usada.

Dejamos el análisis a fondo de estos circuitos para el lector interesado y lo remitimos a las referencias bibliográficas, en especial al libro ***TTL Cookbook***.

7.9 Resumen

Se introduce en este capítulo el primer elemento lógico de construcción de circuitos digitales: el Flip−flop.

Un biestable, además de muchas otras funciones, tiene la propiedad de conservar el estado anterior de las entradas y es también una memoria (limitada a un bit).

Se analizan varios tipos de circuitos biestables y la conveniencia de una señal extra llamada reloj para sincronizar toda la operación.

Se introducen los conceptos de diagrama de tiempos y cómo nos pueden ayudar a comprender el funcionamiento de un circuito. Se analiza la conveniencia de responder a la transición negativa (de 1 a 0) del reloj en lugar de a la positiva (de 0 a 1)

Este circuito lógico nos servirá a lo largo de los demás capítulos como bloque constitutivo de otros circuitos más complejos además de tener múltiples aplicaciones.

7.9.1 Puntos Importantes del Capítulo

- El flip−flop es un elemento de memoria.
- Las compuertas pueden ser analizadas de acuerdo con sus entradas que habilitan o deshabilitan al circuito para continuar realizando operaciones lógicas.
- El flip−flop ***SR*** cambia su estado de acuerdo con las entradas conservando una “memoria” de sus entradas anteriores.
- El reloj es el mecanismo de sincronización.

- A los eventos que dependen del reloj para funcionar se les conoce como síncronos a los que no dependen de él se les llama asíncronos (no síncronos).
- Un diagrama que muestra las entradas a un circuito, sus salidas y cómo responde a los cambios del reloj, se denomina diagrama de tiempos o cronograma y es una herramienta indispensable para analizar circuitos complejos.
- Los circuitos pueden responder ya sea al borde positivo o negativo del reloj.
- La interconexión maestro–esclavo permite que los biestables respondan al borde negativo del reloj.
- El flip–flop ***JK*** no tiene estados inválidos como es el caso de los *SR*.
- El flip–flop ***D*** retrasa la señal presente a su entrada por el tiempo de un ciclo de reloj.
- Un flip_flop ***T*** divide la frecuencia de su entrada.
- Las aplicaciones de los biestables son varias y entre otras están: registros de corrimiento, secuenciadores, eliminadores de ruidos, sincronizadores, memorias, etc.

7.10 Ejercicios

7.1 ¿Podría describir la diferencia de funcionamiento entre el circuito de la figura *7.11a* y el de la figura *7.11b*?

Dibuje sus respectivos diagramas de tiempos sin olvidar incluir la señal del reloj.

Muestre la tabla de verdad para todos los estados posibles en ambos casos.

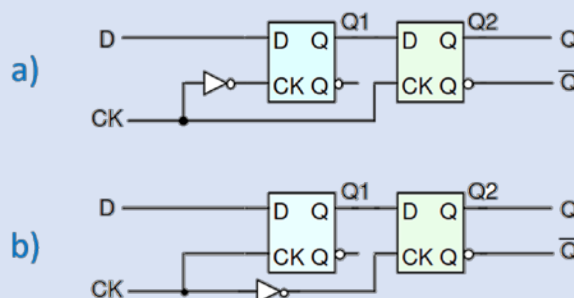


Figura 7.79 Biestables tipo D modificados.

7.2 Liste el estado de salida de un biestable SR para las entradas mostradas en la siguiente tabla (auxíliese de la tabla de verdad de la figura *7.1*):

S	R	Q
0	**1**	Limpia
1	**1**	No permitido
1	**0**	...
1	**1**	
1	**0**	
1	**1**	
0	**1**	
1	**1**	
0	**0**	
1	**0**	

8 Elementos Lógicos Registros y Contadores

8.1 Registro de Corrimiento

Como se estudió en el capítulo anterior, el biestable es un dispositivo de almacenaje, pero de un sólo bit (dígito binario) por lo que se le conoce como *registro de 1 bit*. Si requerimos que ***N*** bits sean registrados o recordados, necesitamos de ***N*** flip-flops. Cuando un arreglo de biestables tiene un número ***N*** de bits almacenados, es necesario algunas veces mover o correr los bits de un biestable a otro de una forma que a continuación describimos. A tal arreglo de flip-flops se le conoce como ***registro de corrimiento*** (shift register).

La operación llamada de corrimiento, es aquella en la que los dígitos almacenados en un registro cambian su posición. Existen dos tipos definidos de corrimiento: A la derecha y a la izquierda. Un corrimiento a la izquierda mueve cada bit de información guardada hacia la izquierda un número específico de posiciones. Un corrimiento a la derecha realiza la misma operación, pero hacia el otro lado. Si tenemos un registro que contiene la información 1011, y lo corremos a la izquierda una posición, tendremos como resultado 0110. Si la operación se realiza a la derecha tendremos 0101.

La figura *8.1a* muestra un registro simple de corrimiento de 4 bits construido con flip-flops del tipo ***D***. También se pueden substituir los biestables ***D*** con ***JK*** conectando un inversor a la entrada del primero de ellos en K_0 como se muestra en la figura *8.1b*. En las siguientes etapas el inversor no se requiere pues se usa $\overline{Q_n}$ como entrada de K_{n+1}. Se pueden substituir los flip-flops ***JK*** con biestables ***SR***.

Recordemos que la característica de un biestable tipo ***D*** es que, inmediatamente después de la transición de disparo del reloj, la salida ***Q*** cambia al estado presente en la entrada ***D*** justo antes de dicha transición. Por lo que cualquier patrón de bits presente a la entrada del flip-flop se desplaza a la derecha con cada cambio del reloj. La salida del último de la cadena se pierde.

El diagrama de tiempo de la figura *8.1c* muestra dicho comportamiento. En dicho diagrama definimos $D_0=0$ y $Q_0=1$ y $Q_1 = Q_2 = Q_3 =$

0, Definimos arbitrariamente que los flip-flops se disparan con el borde negativo del reloj.

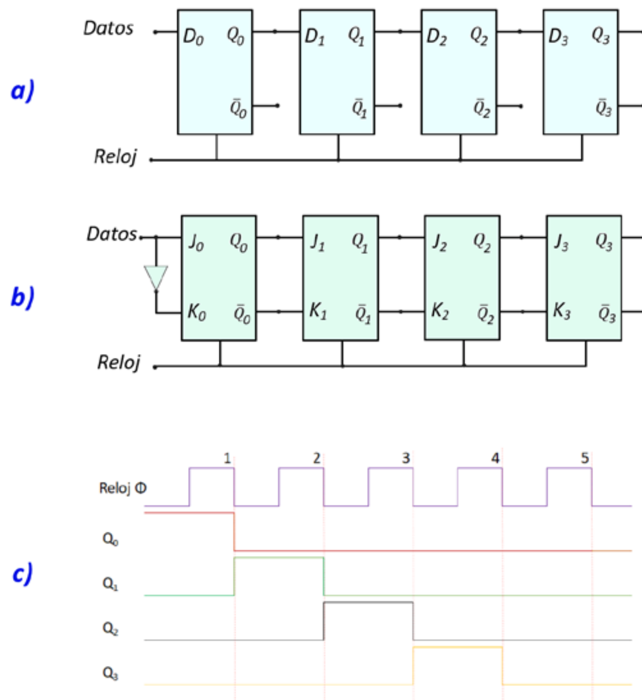


Figura 8.80 Registro de corrimiento de 4 bits.

La configuración del circuito es tal, que la entrada de un flip-flop depende del que lo antecede (excepto el primero). Se pueden agregar o quitar flip-flops si se necesitan más o menos bits. La salida del flip-flop $\boldsymbol{D_k}$ es cero si el biestable que lo precede tiene su salida en 0 (estado estable) con $\boldsymbol{D_{k-1}}$ =0, de la misma forma D_k =1 si Q_{k-1} =1. Los datos del primer flip-flop se determinan de una fuente externa.

En la figura *8.2b* mostramos el diagrama de tiempos de una secuencia 11010 que entra a nuestro registro de corrimiento. Se muestra también la salida de cada uno de los flip-flops a la transición del reloj (recuerde que estamos tratando con biestables que reaccionan a la caída del reloj).

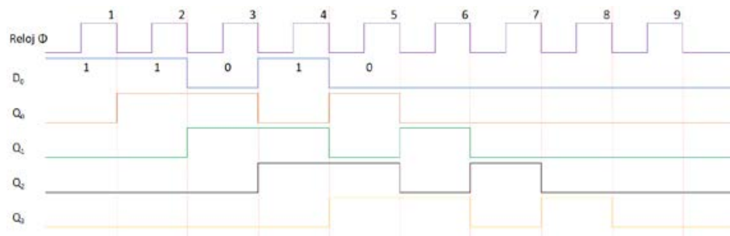


Figura 8.81 Diagrama de tiempos de un registro de corrimiento.

8.1.1 Reloj

Debemos hacer énfasis en que el registro de corrimiento descrito anteriormente opera de esa forma sólo si el cambio del reloj deshabilita al circuito para responder a una entrada que cambia. Después del cambio del reloj, el flip-flop kesimo está determinado por el estado (*k*−*1*) antes del cambio del reloj.

Si se refiere a la figura *7.6a*, donde se presentó por primera vez esta necesidad, reconoceremos ahora el circuito como un registro de corrimiento de 2 bits.

8.1.2 Transferencia Paralela-Serie

La información presente en las líneas de comunicación puede tomar dos formas:

- Serial. Los datos se presentan uno a uno, un bit a la vez, usualmente en sincronía con el reloj.
- Paralelos. Los datos llegan todos a la vez en líneas independientes, tantas como el número de bits que se necesiten.

Un registro de corrimiento es un método común de transferir de una forma a otra de comunicación. Para transferir ***N*** bits de su forma serial a paralela, permitimos que estos ***N*** bits sean la entrada a un registro de corrimiento de ***N*** bits. ***N*** ciclos de reloj después, tenemos las salidas de los flip-flops con los datos seriales y podemos tomar una línea de salida de la terminal ***Q*** de cada flip-flop donde tendremos los datos en paralelo.

De forma similar, podemos usar las entradas ***LIMPIA*** y ***FIJA*** (clear y preset) de cada flip-flop (no representadas en la figura *8.1*) para colocar en un estado inicial conocido a cada circuito (entrada en paralelo). Una vez realizada esta operación podemos iniciar el funcionamiento del registro de corrimiento y tener después de ***N*** ciclos de reloj la salida serial completa.

8.1.3 Acarreo (End Around Carry)

En el registro de corrimiento de la figura *8.1a*, la salida del último flip-flop se pierde. Cuando es necesario preservar los bits guardados en un registro, podemos acoplar la salida del último de la cadena al primero. En tal registro, los bits circularán alrededor del circuito moviéndose a la derecha una posición en cada pulso de reloj. A tal circuito se le denomina registro de corrimiento con acarreo o contador en anillo (ring counter).

Los datos en el registro pueden entrar en forma serial por lo que es necesario proveer un mecanismo que detenga la entrada de datos cuando el registro se encuentra lleno y que haga la conexión del acarreo para comenzar el conteo en anillo. De forma alternativa, los bits pueden ser introducidos de manera no síncrona por medio de las conexiones ***LIMPIA*** y ***FIJA***.

Esta forma de corrimiento tiene muchas aplicaciones dentro de la ***Unidad Aritmética y Lógica*** (***UAL*** o en inglés Arithmetic and Logic Unit ***ALU***) . Recordemos que para multiplicar por la base 2 (la que se usa en sistemas digitales), basta con recorrer a la izquierda una posición el byte que se quiere multiplicar y para dividir se requiere realizar corrimientos a la derecha.

8.1.4 Registro de Corrimiento a la Izquierda y Derecha

Debido a la característica especial de la multiplicación y división de los números binarios, existen ocasiones en que deseamos realizar el corrimiento a la derecha o a la izquierda. Un circuito que realiza esta función se muestra en la figura *8.3*. Los datos que se correrán a la derecha entran por la conexión marcada D_d, mientras los que se corren a la izquierda lo hacen por D_i. La línea ***M*** es la que indica la forma de corrimiento: *M=1* es a la derecha y *M=0* a la izquierda.

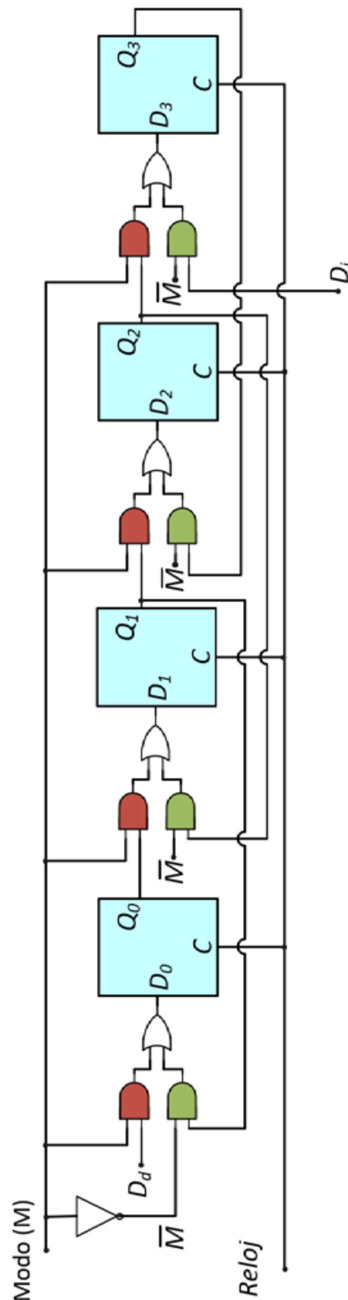


Figura 8.82 Registro de corrimiento a la izquierda y a la derecha.

Cuando *M=1* todas las compuertas indicadas en rojo en la figura están habilitadas, mientras que las indicadas en verde se encuentran desactivadas y el circuito funciona tal como el de la figura *8.1a*. Cuando *M=0* las conexiones están al revés. La entrada ***M*** sólo debe cambiar cuando el reloj esté en cero o el circuito no funcionará adecuadamente.

8.2 Contadores

El flip-flop tiene dos estados, de forma correspondiente un arreglo de ***N*** biestables tiene *2^N* estados, siendo un estado del arreglo definido como una combinación particular de los estados individuales de los flip-flops. Un contador de biestables es un arreglo en el que los circuitos se interconectan de tal manera que el arreglo avance de estado a estado con cada ciclo de la señal de reloj. Si comenzando en un estado inicial, el contador regresa a él después de k ciclos de la señal de entrada, el contador se llama contador de base *k* o contador de módulo *k*. Si el contador cuenta con *N* flip-flops y avanza por cada uno de los estados (característica no siempre deseable), el contador es de módulo *2^N*.

En la figura *8.4* se muestra un contador *modo 16* (*2^4*). Cuatro biestables tipo ***JK*** se conectan de forma que cambian continuamente de estado y sólo el primero de ellos recibe la señal del reloj. Hemos comenzado en un estado arbitrario (al que se puede llegar con las conexiones de ***LIMPIA*** y ***FIJA***) de $Q_3 = Q_2 = Q_1 = Q_0 = 0$ y podemos verificar que la cuenta *k* del circuito se encuentra por:

$$k = 2^3 Q_3 + 2^2 Q_2 + 2^1 Q_1 + 2^0 Q_0 \tag{8.1}$$

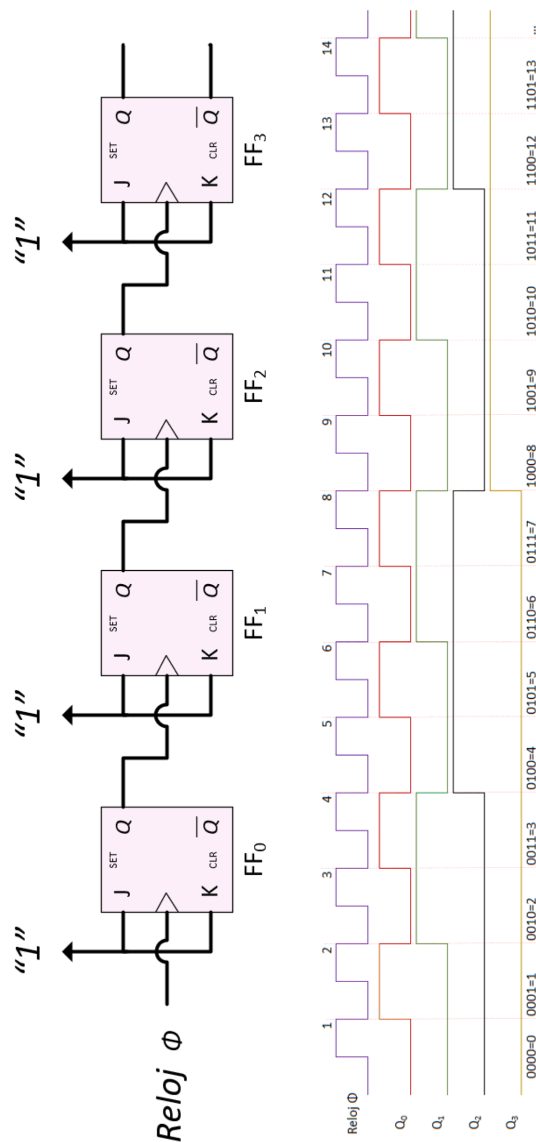


Figura 8.83 Contador módulo 16.

La cuenta del contador puede ser "leída" en forma decimal por medio de un decodificador como el que se muestra en la figura *8.5*. Hemos restringido el número de flip-flops a 3 para mantener las conexiones simples y fáciles de seguir. Como tenemos 3 flip-flop el contador es de módulo 8 (2^3).

Decodificador

Circuito que puede tener varios estados arbitrarios a la entrada y sólo genera un 1 lógico en una de sus líneas de salida mientras que las demás se mantienen en 0 lógico. Normalmente se clasifica por sus líneas de entradas con respecto a las de salida por ejemplo de 2 a 4 o de 3 a 8. Las líneas de salida son necesariamente función de sus N entradas y pueden ser tantas como 2^N.

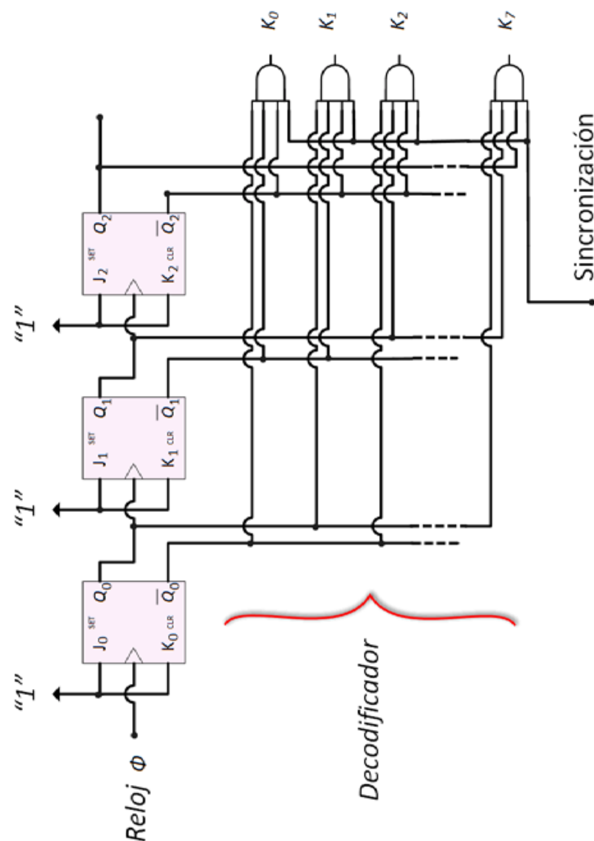


Figura 8.84 Convertidor binario a decimal (decodificador).

Las salidas $k_0, k_1, k_2, \cdots\ k_7$ son 1 sólo en el caso en que su estado correspondiente se refleje en el contador. Por ejemplo $k_0 = 1$ si $Q_2Q_1Q_0 = 000$ y todas las demás salidas son cero $(k_1, k_2, k_3, \cdots\ k_7)$.

Es necesario una señal de sincronización o estroboscopio (strobe) para armonizar el funcionamiento del decodificador pues hay que recordar que todos los circuitos tienen un retraso finito por muy pequeño que éste sea. Normalmente esta señal está en 0 lógico y todas las salidas se encuentran por consecuencia en 0 lógico. Si queremos leer la salida ponemos la señal de sincronización temporalmente en 1 lógico.

A este tipo de contadores se les conoce como ***contadores asíncronos*** pues la señal de reloj sólo se conecta al primero de ellos y los demás no lo comparten. Si por el contrario la señal del reloj fuese compartida por todos los circuitos, el contador sería ***síncrono*** y es una forma de mejorar el funcionamiento de los contadores (ver la figura *8.6* y la bibliografía para más información).

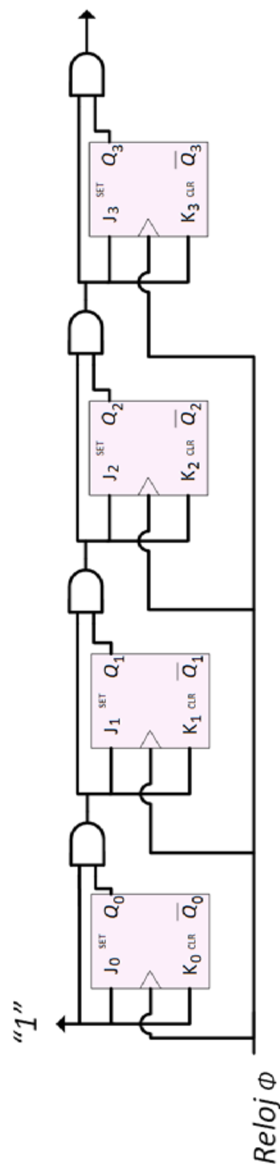


Figura 8.85 Contador serial síncrono.

Debido a la característica no síncrona, hay una frecuencia máxima limitante para la señal del reloj que, si se excede, ocasionara problemas a la salida. Supongamos que el tiempo de propagación de la señal de cada flip-flop es de 20ns, para que el último biestable de la cadena mostrado en la figura *8.4* logre cambiar, deben pasar 4· 20ns = 100ns. Si el intervalo de cambio de reloj es menor a 100ns (una frecuencia de $f = \frac{1}{T} = \frac{1}{100{\cdot}10^{-9}} = \frac{1}{10^{-7}} = 10^7 = 10\ MHz$) el último flip-flop no logra cambiar y la cuenta se pierde.

Muchas veces el circuito de la figura *8.4* no se usa para contar sino para dividir la frecuencia del reloj en una cantidad fija como podemos ver en el diagrama de tiempos de la figura *8.4*. En dicho diagrama observamos que Q_0 divide a la mitad la frecuencia de entrada, Q_q en la cuarta parte y en general la salida de un contador de *N* etapas divide la entrada por un factor de 2^N. Usado de este modo, al circuito se le llama ***divisor de frecuencia***.

Ejercicio

8.1 Diseñe un decodificador de 4 líneas para un contador de módulo 4 (con dos biestables). Indique con claridad cada paso para llegar al diseño del contador y a la simplificación del decodificador.

8.2.1 Contadores no binarios

Hasta el momento sólo hemos visto contadores que después de pasar por todos los estados regresan a la cuenta inicial, esto es, son potencias de 2. Es deseable muchas veces utilizar un contador no binario, por ejemplo, un contador de base 10 es de especial interés. Analizaremos ahora cómo un circuito de contador puede ser modificado para contar en un módulo arbitrario que no sea potencia de 2.

Suponga que se quiere diseñar un contador módulo *k*. Se debe comenzar con *N* flip-flops de forma que 2^N sea menor que *N*, por ejemplo, para un contador $k=5$, 6 o 7 se necesitan 3 flip-flops pues $2^3=8$. Para $k=9$ hasta 15 usamos $N=4$, etc. Para construir el contador, arreglamos los biestables de forma que algunos de los estados por los que podría pasar sean omitidos y vuelva a la cuenta inicial. Para un contador de módulo *k* debemos omitir 2^N-k estados. Tenemos la libertad de decidir qué estados omitimos por lo que hay muchas formas de diseñar un contador. Un contador puede ser usado para manejar un despliegue numérico o para controlar la secuencia de ciertos eventos, pero en casi todos los casos debe existir un decodificador a la salida del circuito. Usualmente las decisiones con respecto a qué estados no usar son realizadas en interés de la simplificación del circuito resultante.

Diseñaremos como ejemplo un contador módulo 3. Un contador módulo 3 requiere 2 flip-flops. En la figura *8.7a* se especifica la secuencia deseada de la cuenta. Si usamos biestables tipo ***JK*** es necesario presentar la tabla de transiciones de una nueva forma

(vea la figura *8.7b*) en donde tomemos en consideración el estado de la salida en un momento ***n*** y luego en el momento ***n+1***. Las ***X*** representan estados de no importa.

a)

edo. del contador	Q_0	Q_1
0	0	0
1	0	1
2	1	0
0	0	0
...	...	...

b)

Q_n	Q_{n-1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

c)

J_0:

D_1 \ D_0	0	1
0	1	X
1	0	X

K_0:

D_1 \ D_0	0	1
0	X	1
1	X	X

J_1:

D_1 \ D_0	0	1
0	0	1
1	X	X

K_1:

D_1 \ D_0	0	1
0	X	X
1	1	X

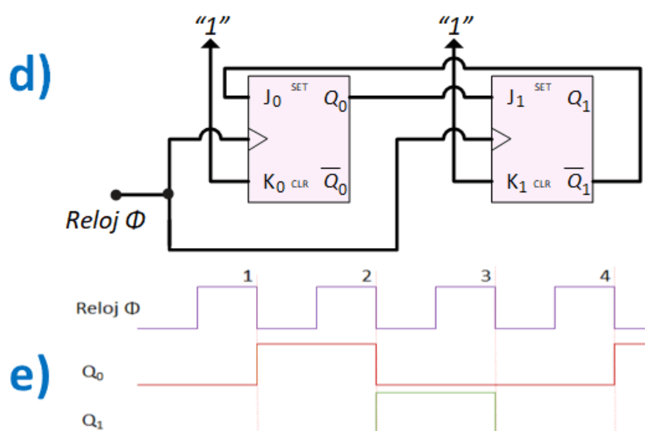


Figura 8.86 Diseño de un contador módulo 3.

Usando como referencias las tablas de la figura *8.7a* y *8.7b* podemos proceder a llenar los mapas de Karnaugh de la figura *8.7c* de la siguiente forma:

Para el estado $Q_1Q_0 = 000$ la siguiente transición nos lleva al estado *01* donde Q_1 no cambia (estado $K_1 = 0,\ J_1 = 1$ de la tabla de la figura *8.7b*) mientras que Q_0 cambia de estado a 1 lógico ($K_0 = 1,\ J_1 = X$ de la tabla de la figura *8.7b*) con lo que podemos llenar los mapas en la posición *00* con *0X* y *1X*. De la misma forma tenemos:

posición 01: 1X y X1
posición 10: X1 y 0X

La posición *11* nunca sucede por lo que se llena con *XX.*

De la simplificación de los mapas llegamos a cada una de las entradas de los flip-flops y a la figura *8.7d* cuyo diagrama de tiempos se muestra en la figura *8.7e*.

8.2.1.1 Bloqueo

En el contador módulo 3 diseñado en la sección anterior cuya implementación se muestra en la figura *8.7d*, el estado *11* no se usa. Puede ser que por casualidad el contador se encuentre en cierto momento en ese estado no utilizado. Tal situación puede suceder por causa de ruido externo al circuito y que afecte su estado inicial. En estos casos, el contador puede pasar de un estado no usado a otro y nunca llegar a iniciar la cuenta (el caso es más claro en contadores con muchos estados que no se usan). Si esto llega a suceder, el contador queda inutilizado para su propósito original. Un contador cuyos estados sin uso tienen esta característica se denomina ***contador bloqueado***.

Para asegurarse de que el contador no llegue a estos estados sin uso, ya sea por error o al arrancar el circuito, se utiliza lógica externa que nos permita llevar al contador a un estado inicial estable y conocido. Si el contador inicia en un estado sin uso y, después de pasar por otros estados, llega finalmente a su estado inicial de uso, el circuito puede considerarse como aceptable y tomarse estos estados como "calentamiento" del circuito o errores que deben descartarse de la cuenta inicial. En todo caso, debe verificarse el diseño de un contador para asegurarse que el paso por un estado sin uso no nos lleve a una condición de bloqueo.

8.3 Generadores de Secuencias

Un generador de secuencias es un sistema que, en sincronía con el reloj, genera una secuencia fija de bits lógicos. Tal generador puede usarse como contador, divisor de frecuencia para propósitos de llevar el tiempo, como generador de códigos, para el reconocimiento de secuencias, etc.

La *longitud* de una secuencia es el número de bits sucesivos en la secuencia antes de que el patrón se repita. Por ejemplo, la secuencia 110011001100... tiene una longitud de 4 bits. Puesto que el

generador de secuencias repite el patrón sin descanso, podemos leer la secuencia de números de muchas formas dependiendo del punto de inicio. La secuencia anterior puede ser la repetición del patrón 1100, 0011, 0110, etc. Es interesante resaltar que en muchas aplicaciones la longitud del patrón es más importante que la secuencia exacta de bits.

La estructura básica de un generador de secuencias se muestra en la figura *8.8*. Tenemos *N* flip-flops en cascada en la forma usual de un registro de corrimiento y tiene cierta reminiscencia a la estructura de un contador en anillo. En un contador de anillo, la salida del último circuito es la entrada del primero. En el presente caso, sin embargo, la entrada del primer circuito se forma también por una función de la salida de todos los demás circuitos. Dejamos al lector interesado la búsqueda de los procedimientos de diseño en la bibliografía seleccionada.

Figura 8.87 Estructura básica de un generador de secuencias.

8.3.1 Secuenciadores

En el circuito de la figura *8.9* mostramos un circuito que espera una serie de eventos por su entrada de reloj y nos da a la salida un pulso que cambia de 1 a 0 lógico en caso de que la secuencia se cumpla. Consiste en una serie de flip-flops del tipo ***D*** en cascada. Es usual en circuitos donde se deben predeterminar cuentas, candados (seguros) y alarmas digitales. Para iniciar el funcionamiento se deben usar las entradas ***FIJA*** de forma que la salida ***Q*** de cada circuito se encuentre en 1 lógico. Si el evento ***A*** ocurre, la entrada ***D*** del primer flip-flop que se encuentra en "0" lógico o "tierra" se pasa al segundo biestable; si ***B*** ocurre y sigue a ***A***, el cero pasa a la entrada de la tercera etapa. Finalmente, si ***C*** ocurre y sigue a ***A*** y a ***B***, el cero aparece en la salida. El circuito puede usarse en contadores donde cuando los millares, centenas, decenas y finalmente las unidades alcanzan una cuenta exacta se producen una salida hacia otro circuito. En circuitos de alarma y candados electrónicos, la secuencia correcta debe teclearse para activar el circuito. Si la secuencia correcta no se introduce, se pueden agregar circuitos adicionales para activar una alarma o para comenzar un retraso de tiempo que desactive la entrada. Una variante del circuito se utiliza para la activación de alarmas al salir la última persona por una puerta protegida. El abrir y cerrar la puerta activará la última etapa del circuito para armar la alarma.

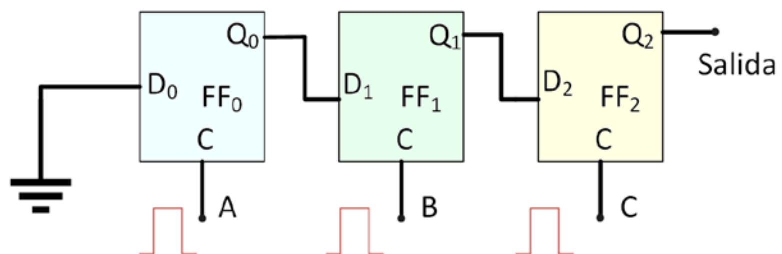


Figura 8.88 Circuito de secuencias.

8.4 Resumen

Los circuitos de corrimiento, contadores y generadores de secuencia forman la parte básica de la *Unidad Aritmética y Lógica* junto con los sumadores (analizados en el siguiente capítulo). Es, por lo tanto, de especial interés el conocimiento de su funcionamiento y diseño elemental.

8.4.1 Puntos Importantes del Capítulo

- Existen registros de corrimiento a la derecha e izquierda.

- El correr un registro a la izquierda o derecha equivale a multiplicar por 2 o dividir entre 2.
- Una aplicación adicional de un registro de corrimiento es como convertidor paralelo−serial o serial−paralelo.
- Un contador es un circuito que, comenzando en un estado inicial, regresa a éste después de ***N*** ciclos de reloj.
- Tenemos dos tipos de circuitos contadores: los síncronos y los asíncronos (no síncronos).
- El bloqueo es una condición que sucede cuando un contador comienza en alguno de los estados no utilizados y no puede salir de ellos.
- Un contador puede ser usado también como divisor de frecuencia.

9 Elementos Lógicos
La Unidad Aritmética y Lógica

9.1 La Unidad Aritmética y Lógica (UAL o ALU)

Los circuitos que a continuación describimos forman, en conjunto, parte de lo que se llama la ***Unidad Aritmética y Lógica***[30] (***UAL***, en inglés ***ALU***) que es parte de la *Unidad de Procesamiento Central* (***UPC***, Central Processing Unit o ***CPU***). La Unidad Aritmética y Lógica es la sección de la computadora que realiza las operaciones matemáticas y lógicas en los datos de entrada procesados por el computador. Esta sección de la máquina puede ser relativamente pequeña consistiendo en quizá uno o dos circuitos de integración a gran escala (***LSI***); formar parte de la propia computadora como en el caso de la microcomputadora o ser una serie considerable de circuitos lógicos de alta velocidad como en las macrocomputadoras o supercomputadoras. No importando el tamaño y la complejidad de estos circuitos, las máquinas pequeñas realizan generalmente las mismas operaciones lógicas y aritméticas, usando los mismos principios que en las grandes máquinas. Lo que cambia es la velocidad de las compuertas lógicas y los flip-flops utilizados; también, técnicas especiales son utilizadas para realizar varias operaciones en paralelo.

Aunque muchas funciones pueden ser realizadas por los ***UAL*** de las máquinas de hoy en día, las operaciones aritméticas básicas (suma, resta, multiplicación y división) continúan siendo las operaciones más utilizadas. Inclusive las especificaciones de una computadora nos dan evidencia de la naturaleza fundamental de estas operaciones: en cada máquina nueva se describen los tiempos requeridos para la suma y multiplicación como características significativas.

Es importante resaltar que existe otra parte de la computadora llamada ***Unidad de Control*** que dirige las operaciones del ***UAL***. Todo lo que hace el ***UAL*** es sumar, restar, acarrear, etc. cuando se

---

[30] Los transistores empleados hoy en día en los circuitos integrados (IC) son órdenes de magnitud más pequeños que los de los primeros microprocesadores. Esto ha permitido integrar una o varias ***UAL*** altamente complejas en los mismos circuitos integrados que albergan la ***UPC***.

le provee de la secuencia correcta de las señales de entrada. Depende del elemento de control el proveer de estas señales, así como es función de la ***unidad de memoria*** proveer a los elementos aritméticos de la información que usarán. Asumiremos por el momento que las secciones de control y memoria de la máquina son capaces de generar las señales de control correctas y que los datos en los cuales se va a realizar la operación, están disponibles.

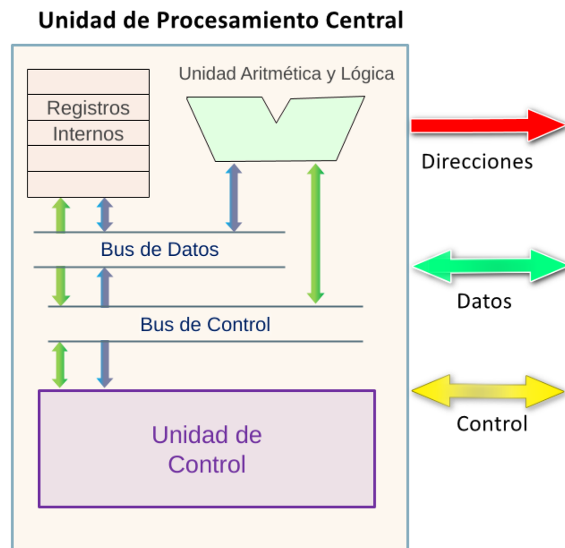


Figura 9.89 La Unidad Aritmética y Lógica.

9.1.1 Construcción del ALU

La información que se maneja dentro de una computadora se divide en palabras, cada una de ellas con un número fijo de bits (algunas computadoras proveen la habilidad de manejar operadores de longitud variable). Por ejemplo, las palabras manejadas por una determinada máquina binaria pueden ser de 32 bits de longitud. Los operadores usados se obtienen de la memoria y la unidad de control dirige las operaciones que se realizarán de acuerdo con un programa determinado. Si se requiere de una suma, los dos sumandos deben proporcionarse a la ***UAL*** que los sumará y luego se debe guardar el resultado, aunque sea en forma temporal.

Para introducir varios conceptos, vamos a considerar la construcción típica de un ***ALU***. Los dispositivos de almacenaje consisten en una serie de ***registros*** de flip-flops, cada uno de ellos puede estar formado de uno o varios flip-flops. La longitud de estos registros se define como la cantidad de información que pueden almacenar. En un registro binario, esto es igual al número máximo de bits

que caben en un registro; en un registro del tipo ***BCD***, al número máximo de dígitos decimales que se pueden almacenar. Por conveniencia a cada uno de los registros dentro de la computadora se le da un nombre tal como registro ***A***, registro ***X***, etc. y a los flip-flops que lo forman se les da el mismo nombre, esto es, el registro ***A*** se forma de los flip-flops $A_0, A_1, A_2, \dots, A_N$. Algunas computadoras tienen uno o varios registros llamados ***acumuladores*** que es el principal registro para las operaciones lógicas y aritméticas. El registro guarda los resultados de las operaciones lógicas y aritméticas; se anexan los circuitos lógicos requeridos a este registro para que las operaciones necesarias puedan realizarse con su contenido y los otros registros que intervengan.

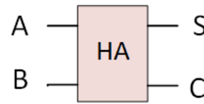
El acumulador es, por lo tanto, un registro básico de almacenamiento para los elementos aritméticos. Si se le dan instrucciones a la máquina de "Cargar" el acumulador, el elemento de control se encargará de limpiar el registro de su contenido anterior y de poner el operando contenido en la memoria en el acumulador (la forma exacta de hacerlo se verá con cierto detalle en los siguientes capítulos). Si se pide ahora la suma, la unidad de control deberá pasar la cantidad del acumulador como uno de los sumandos a la ***UAL*** y el otro sumando tomarlo de una localidad de memoria y regresar el resultado al acumulador o a la memoria. Nótese que, si el contenido se regresa al acumulador, este pierde su contenido original y contiene al final de la operación la suma resultante. En este capítulo trataremos sólo con el proceso de sumar, restar, etc. y no con el proceso de localizar el número a ser sumado de la memoria o de regresar el resultado a la misma. Estas operaciones se describirán en los siguientes capítulos.

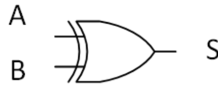
Algunas computadoras tienen más de un acumulador y se denominan acumulador ***A***, acumulador ***B***, etc., o ***ACC***$_1$***, ACC***$_2$***, AX, BX,*** etc. dependiendo del computador o microcomputador analizado. Cuando el número de registros previsto para retener a los operadores se vuelve grande (4 o más) se acostumbra a nombrarlos registros generales.

9.2 La Suma de dos Números

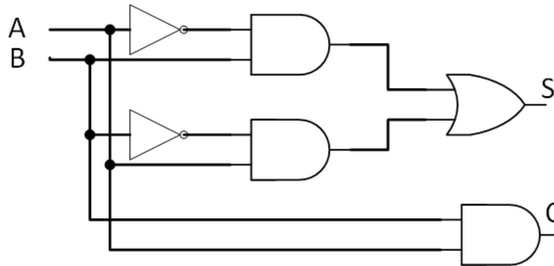
El módulo básico usado en los elementos aritméticos es el ***medio sumador*** (Half adder). La función de un medio sumador es sumar dos dígitos binarios y producir una suma (***S***) y un acarreo (***C***) de acuerdo con las reglas de suma mostradas en la tabla de la figura *9.2*. En la misma figura se muestra el diseño de un medio sumador.

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1





$S = A \oplus B = A\bar{B} + \bar{A}B$



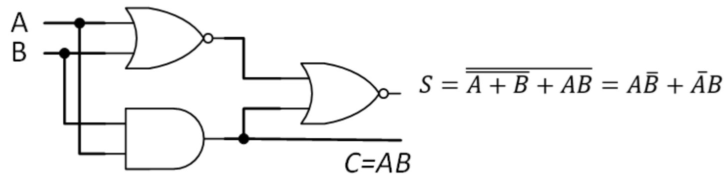


Figura 9.90 Medio sumador.

El dígito de acarreo se relaciona a los dos sumandos por la lógica de la compuerta ***Y***:

$$C = A \cdot B \quad (9.1)$$

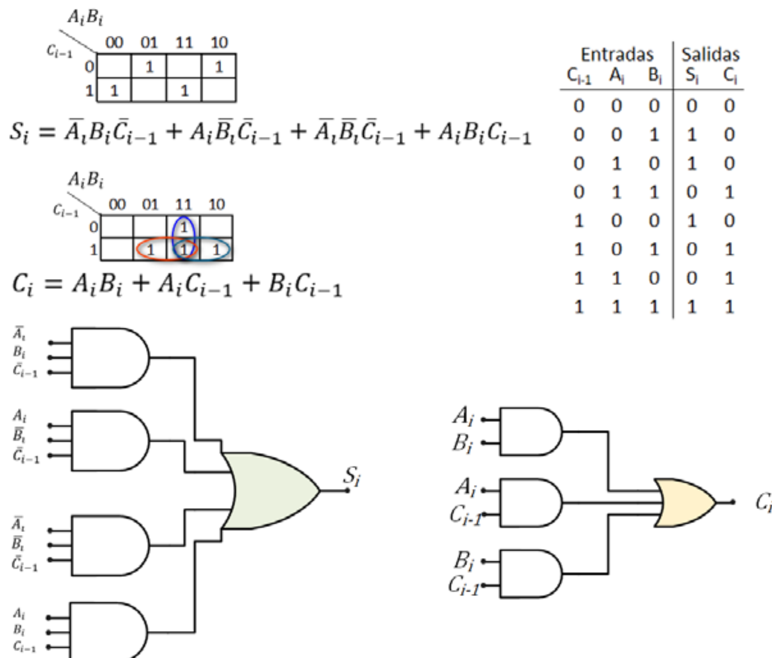
mientras que la suma se da por la función ***O EXCLUSIVA***:

$$S = \bar{A} \cdot B + A \cdot \bar{B} = A \oplus B \quad (9.2)$$

Algunas veces se habla de un ***cuarto de sumador*** que consiste en el mismo circuito que para el medio sumador, pero sin considerar el acarreo. La suma no es completa sin considerar el posible acarreo resultante de la suma de dos bits anteriores. Una compuerta que realiza esto se conoce con el nombre de sumador completo y puede construirse a partir de dos medios sumadores (de ahí su nombre)

9.2.1 Sumador Completo

Cuando más de dos dígitos binarios se van a sumar, varios medios sumadores no son adecuados puesto que el medio sumador no tiene entrada para manejar el acarreo. En la figura *9.3* mostramos la tabla de verdad que un sumador completo debe satisfacer; en la misma figura aparecen los mapas ***K*** y la realización con dos niveles de compuerta para el bit de suma y el bit de acarreo.



Entradas			Salidas	
C_{i-1}	A_i	B_i	S_i	C_i
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Figura 9.91 Sumador completo.

Existen varias otras estructuras de compuertas aparte de las dadas en la figura *9.3* con las que un sumador completo puede realizarse. Por ejemplo, otra de las formas en que S_i y C_i pueden generarse son:

(9.3) $$S_i = C_{i-1} \oplus A_i \oplus B_i$$

(9.4) $$C_i = A_iB_i + (A_i \oplus B_i)C_{i-1}$$

En la figura *9.4* mostramos cómo un sumador completo puede realizarse utilizando dos medios sumadores. Un medio sumador es usado para combinar los dos dígitos de entrada y dar el dígito de suma S_i' y un acarreo C_i'. El segundo medio sumador combina al tercer dígito con los primeros dos. El símbolo del circuito para un sumador completo se muestra en la misma figura.

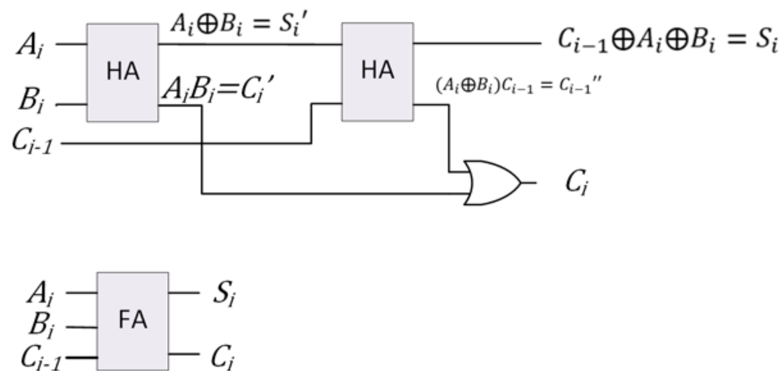


Figura 9.92 Sumador completo realizado con medios sumadores.

9.2.2 Sumador en Serie

Un sumador en serie es un circuito donde dos números ***A, B*** se suman un bit a la vez, comenzando por el dígito menos significativo. El circuito está formado por 3 registros de corrimiento, un sumador completo y un flip-flop tipo ***D*** (ver la figura *9.5*). Para ver cómo opera un sumador serial vamos a asumir que con el último ciclo de reloj hemos cargado los dos números a sumar de ***N*** bits en los dos registros de corrimiento. Asumimos también que el registro de corrimiento corre hacia la derecha y que el estado inicial del flip-flop está en *Q=0*. Eventualmente la suma aparecerá en el registro de corrimiento de *N+1* bits. Durante el primer ciclo de reloj la siguiente serie de eventos tiene lugar.

1. Se transfiere la suma de los dos primeros bits de los sumandos (que estaban presentes en los últimos flip-flop de cada uno de los registros de corrimiento) al biestable más a la izquierda del registro de corrimiento de la suma.
2. Se corre el contenido de los registros de suma un bit a la derecha.
3. Se transfiere la salida del flip-flop ***D*** a la entrada del sumador completo.

Uno a uno, cada bit de los sumandos se procesa hasta que en el pulso de reloj ***N+1*** se transfiere el último bit al registro de suma y el proceso se termina.

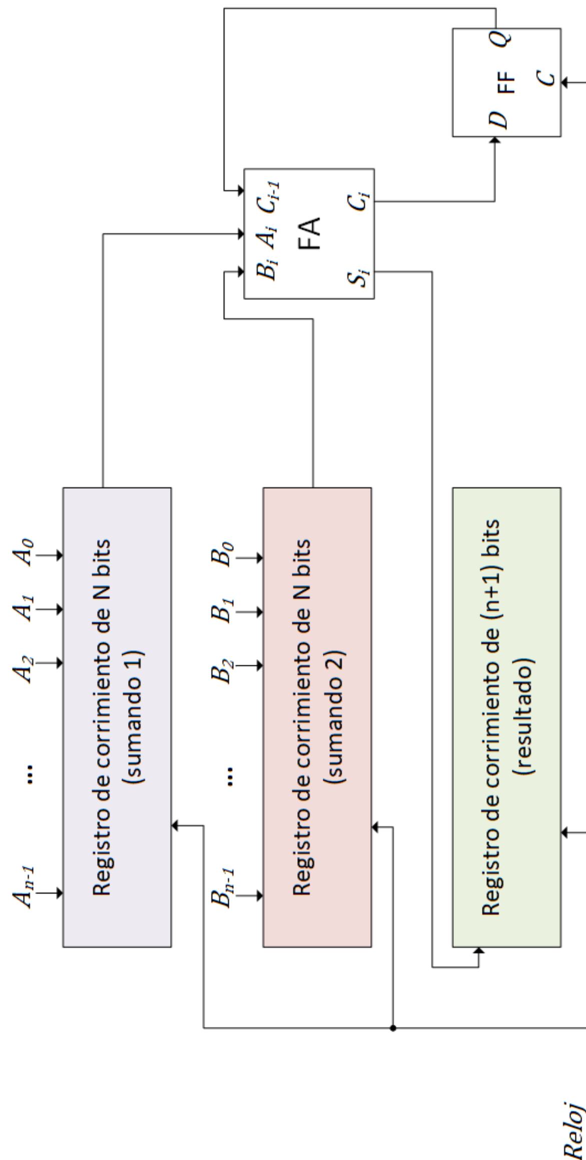


Figura 9.93 Sumador en serie.

El funcionamiento de un sumador en serie nos deja con muchas preguntas sin respuesta:

- ¿Dónde se mantenían los dos sumandos mientras se transferían a sus respectivos registros?
- ¿Cómo se cargan estos registros?
- ¿Cómo se desactiva el sumador hasta que la carga esté completa?
- ¿Cómo sabe el sumador que la suma se ha completado?

- ¿Cómo se limpia el flip-flop ***D*** antes de comenzar la operación?
- Etc.

Las respuestas a estas preguntas nos darán pie a otras más hasta que finalmente hayamos llegado a los requerimientos de entrada y salida del equipo (memoria, teclado, despliegue, impresión, etc.) por los que físicamente se introduce un problema a un sistema digital y finalmente se lee su respuesta. Le corresponde a un diseñador de sistemas digitales dar respuesta a todas estas preguntas. Por el momento sólo nos importa el proceso por el cual la suma se evalúa.

9.2.3 Suma en paralelo

En la suma en serie, la suma es similar a la acción que realiza un humano al calcular, esto es, se va sumando dígito a dígito una columna a la vez hasta llegar al dígito más significativo. Este método es económico en el uso de la electrónica, pero es relativamente lento puesto que cada par de bits sumados requiere un ciclo de reloj. La suma en paralelo en la que todos los bits de los sumandos se adicionan simultáneamente es mucho más rápida que la suma serial, pero con menos economía en el uso de compuertas, puesto que se requiere un sumador por cada par de bits a sumar (ver la figura *9.6*).

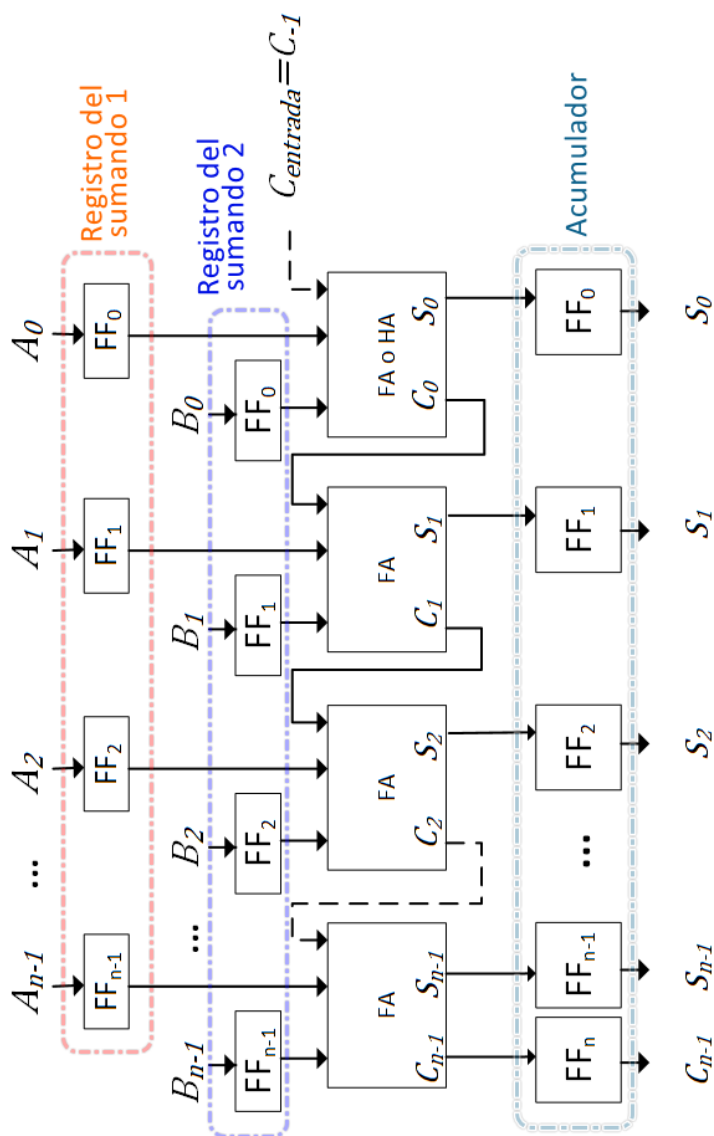


Figura 9.94 Sumador en paralelo.

Cada sumando está contenido en un registro formado por flip-flops y requerimos tantos biestables como dígitos haya en el sumando. Los sumandos pueden haberse almacenado en algún dispositivo de memoria y sus bits dirigidos al flip-flop apropiado por alguna estructura de compuertas (no mostrada en la figura *9.6*). La ventaja de un registro local para cada uno de los sumandos es que una vez que el número a sumar ha sido transferido al registro, la memoria y las compuertas utilizadas para efectuar la transferencia están disponibles para otros propósitos. Nótese que los registros no son registros de corrimiento sino registros de almacenamiento. Puesto que no hay una secuencia serial a seguir, no hay

necesidad de correr bits a través de la longitud del registro y los flip-flops son enteramente independientes uno del otro.

9.2.4 Sumadores Rápidos

Los sumadores analizados hasta el momento sufren de las consecuencias de pasar la información de un sumador a otro (efecto de ola u onda). Existen técnicas para mejorar la velocidad de respuesta de estos circuitos y una de ellas consiste en agregar al sumador un circuito que se encargue de verificar si el dígito de acarreo necesita sumarse en la siguiente columna o no. Esto, que a primera vista parece ser más engorroso, resulta en un aumento de la velocidad neta de la suma, que es uno de los circuitos más usados dentro de la estructura general de una computadora (cálculo de direcciones, sumas de dos números, desplazamientos por la memoria, etc.)

9.3 Resta

Recordemos las reglas de la resta observando la tabla de la figura *9.7*. Llamaremos a ***A*** el minuendo y a ***B*** el substraendo y a ***D=A−B***, la diferencia siendo ***C*** el acarreo resultante en algunos casos. Cuando ***A=0*** y ***B=1*** necesitaremos pedir prestado de la siguiente columna para permitir que la resta pueda realizarse. Cuando se regresa la unidad que fue prestada, es necesario realizar primero una resta del substraendo del minuendo y luego una resta más de 1 al resultado (el orden en que se realicen estas dos operaciones no tiene importancia).

Una estructura de compuertas que acepte a ***A*** y ***B*** como entrada y nos de la resta ***D*** como resultado es llamado medio restador.

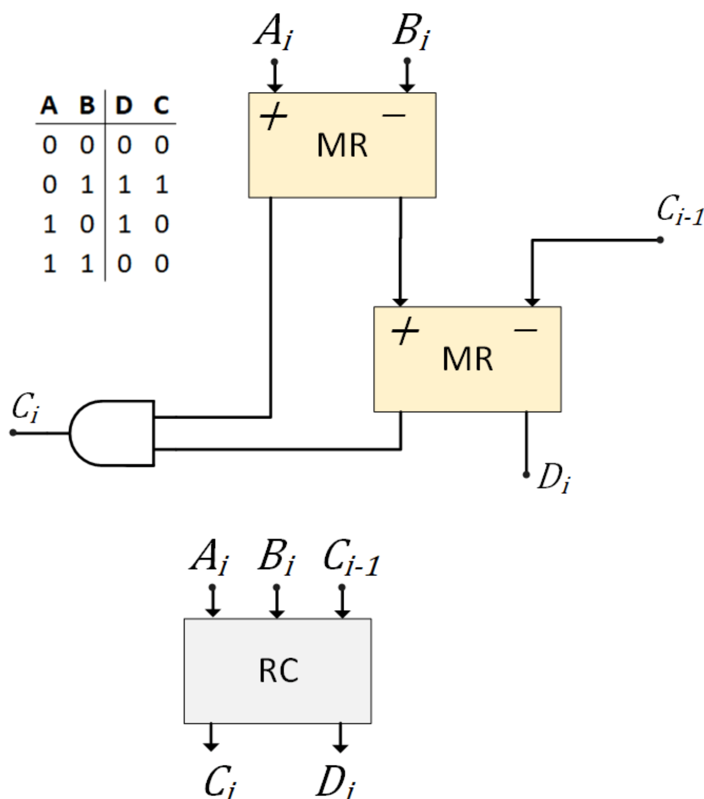


Figura 9.95 La resta.

Podemos verificar que ***D*** y ***C*** están dados por la relación:

(1.16) *(9.5)* $D = A \oplus B \,; C = \bar{A} \cdot B$

Por lo que el circuito resulta ser igual que para un medio sumador, pero el acarreo es distinto.

De forma similar a la suma, un circuito capaz de realizar la resta tomando en cuenta el préstamo anterior, es llamado restador completo. Una de las formas de construir un restador completo es usando dos medios restadores. El sumador completo realiza las dos restas descritas anteriormente y un préstamo del dígito siguiente (si es que lo hay).

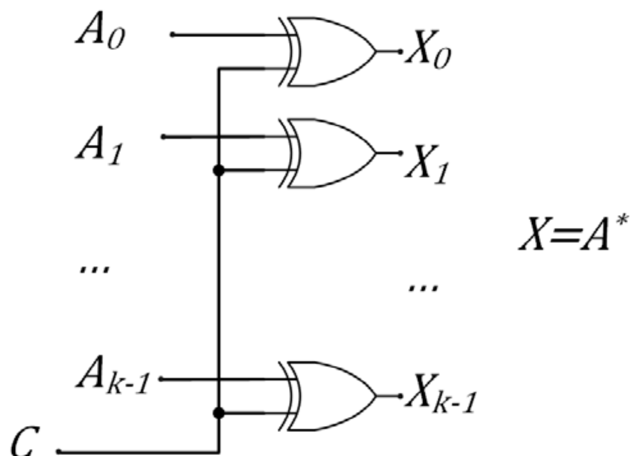
9.4 Números Complementarios

En la sección aritmética de una computadora requerimos tanto de circuitos que puedan realizar la suma como la resta. Como ya vimos en capítulos anteriores, la resta puede lograrse con el complemento del número y sumando en lugar de restar. Cuando la resta se logra de este modo, un restador explícito no es necesario

y usualmente hay un gran ahorro en los circuitos que componen la Unidad Aritmética y Lógica.

Una forma sencilla de encontrar el complemento a unos de un número, sin que para ello tengamos que recurrir a la negación o a la resta, se muestra en la figura *9.8*.

Figura 9.96 Método para generar el complemento a unos.



Cuando la señal de control ***C=0***, a la salida del circuito tenemos el número sin complementar (***X=A***) y cuando ***C=1***, a la salida tenemos el complemento a unos de los dígitos presentes a la entrada ($X = A^*$).

9.5 Multiplicación

Representamos una multiplicación en binario en la figura *9.9*. El multiplicando es multiplicado por cada dígito del multiplicador y estos productos parciales son sumados luego para obtener el resultado total considerando sus posiciones relativas. Cada producto parcial es 0 o igual al multiplicando corrido a la izquierda dependiendo si el multiplicador es 0 o 1. Notamos también que el resultado tiene más dígitos que el multiplicando o el multiplicador y si estos tienen ***N*** bits; el resultado puede llegar a tener hasta $\boldsymbol{2^N}$ bits. Si los registros usados para contener el resultado no tienen la capacidad adecuada, pueden llegar a saturarse y perder el resultado.

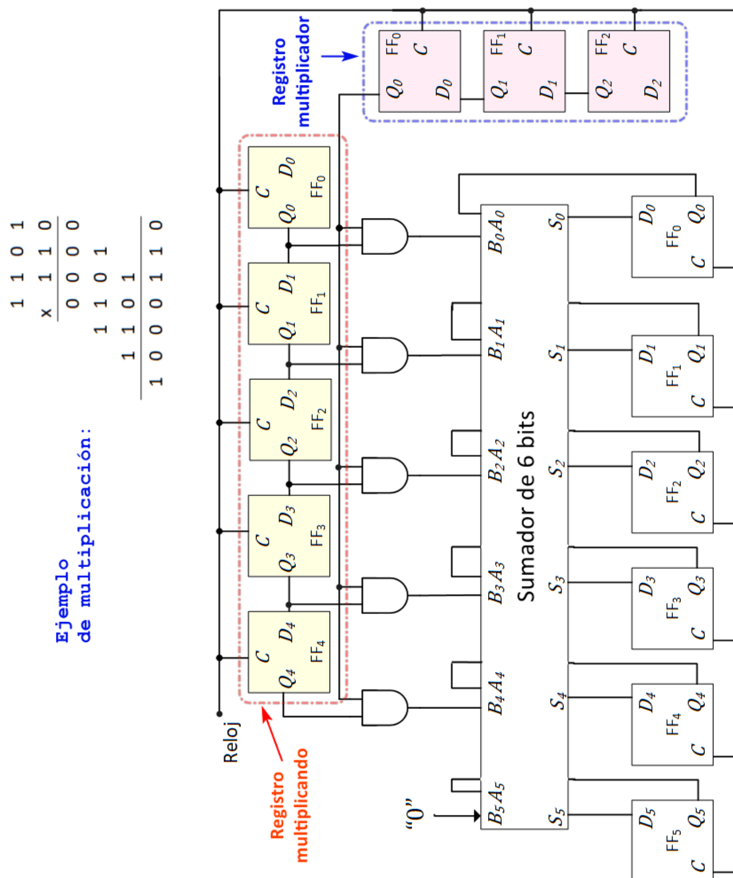


Figura 9.97 La multiplicación.

Un circuito para multiplicar básico para dos palabras de 3 bits se muestra en la figura *9.9*. Consiste en un registro de corrimiento de 5 bits para el multiplicando, uno de 3 bits para el multiplicador y un acumulador (registro totalizador) para el resultado formado con biestables del tipo ***D***. Inicialmente el multiplicando se carga en los tres biestables de la derecha del registro de corrimiento; el dígito menos significativo en el FF_0 y el más significativo en el FF_2. El acumulador se encuentra limpio.

El multiplicador serial presentado es relativamente lento pues requiere que las sumas se realicen una detrás de la otra, por lo que se prefiere usar multiplicadores paralelos que realizan las sumas a la vez y disminuyen el tiempo total de la operación.

9.6 División

La división se puede realizar por medio de restas consecutivas. Puesto que es un proceso de prueba y error, tarda mucho tiempo

en realizarse y últimamente se ha preferido hacer un circuito auxiliar o paralelo a la propia computadora que realice toda la serie de operaciones trascendentes que ocupan la mayor parte del tiempo de la máquina central (coprocesador o procesador numérico).

Presentamos en la figura *9.10* un circuito que realiza la operación de dividir. El dividendo *X* se carga inicialmente en el registro de 4 bits formado por los biestables del 0 al 3 indicados en la parte inferior de la figura. La salida de los registros se conecta como una entrada del sumador, mientras que la otra lo forma el complemento a unos del divisor ***Y***. En principio, el último acarreo debería conectarse al primer acarreo, pero sólo estamos interesados en esa conexión mientras C_3 sea 1, por lo que se prefiere mantener el acarreo inicial en 1. El contador se limpia inicialmente para tener una cuenta de 0.

Si suponemos que $X>Y$, $C_3 =1$ y que la compuerta ***Y*** se encuentra habilitada, entonces la diferencia $X−Y$ aparecerá en $S_3, S_2, S_1\ y\ S_o$. Si cerramos el interruptor que controla la entrada del reloj, pasaremos la diferencia $X−Y$ al registro siguiente y avanzaremos el contador una posición. Tenemos ahora, a la salida del sumador, $(X - Y) - Y = X - 2Y$ en $S_3, S_2, S_1\ y\ S_o$. Si C_3 es aún 1, la compuerta ***Y*** está habilitada y un segundo pulso del reloj carga el contador con $X - 2Y$ avanzando el contador un estado más, etc. Después de ***N*** pulsos de reloj, cuando el residuo es menor que ***Y***, tendremos que $C_3 =0$ y la compuerta ***Y*** queda deshabilitada deteniendo el contador. En el contador tendremos el cociente de la división.

La explicación más detallada de su funcionamiento lo dejamos al lector interesado para consultar en la bibliografía u otras fuentes similares.

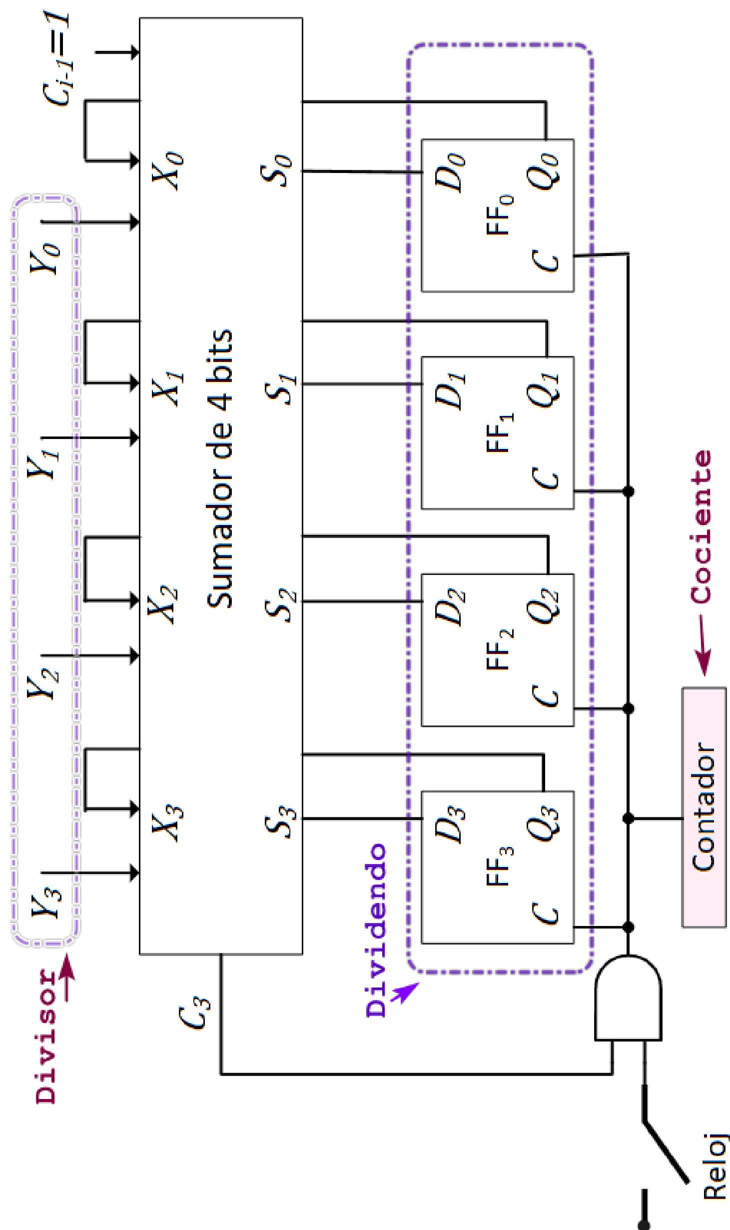


Figura 9.98 La división.

9.7 Ejemplo Comercial de Unidad Aritmética y Lógica

Existen ***UAL*** de las distintas familias que se venden como unidades separadas especializadas en su función. En las microcomputadoras actuales, la ***UAL*** forma parte del mismo circuito y no se puede usar por separado. Un circuito de ***UAL*** es muy versátil y permite realizar en un sólo paquete muchas de las operaciones aritméticas y lógicas.

En la figura *9.11* presentamos un circuito funcional de la compañía Motorola designado como *MC10181*[31] de 24 patillas ***DIP*** que acepta dos palabras de 4 bits y un acarreo junto con 4 líneas que especifican la función a realizarse dentro de la ***UAL***. Consta también de dos terminales más que permiten conectar al circuito con otros circuitos para construir palabras más grandes o para mejorar su velocidad. Las funciones listadas en la tabla de la ***UAL*** son engañosas pues los fabricantes decidieron poner *A menos B menos 1* en lugar de ***AB.***

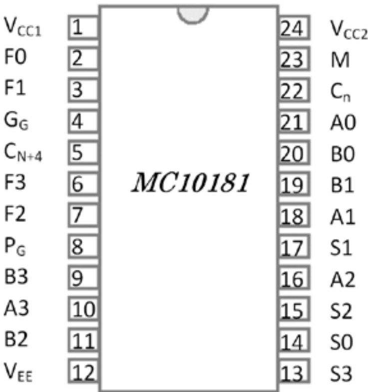


Figura 9.99 Ejemplo de una Unidad Aritmética y Lógica comercial.

Función seleccionada				Función Lógica	Operación Aritmética
				M=1	M=0; C_n=0
S3	**S2**	**S1**	**S0**	***F***	***F***
0	0	0	0	$\bar{A}$	A
0	0	0	1	$\bar{A}+\bar{B}$	$A\ más\ (A\cdot\bar{B})$
0	0	1	0	$\bar{A}+B$	$A\ más\ (A\cdot B)$
0	0	1	1	$"1"\ lógico$	$A\ x\ 2$
0	1	0	0	$\bar{A}\cdot\bar{B}$	$(A+B)\ más\ 0$
0	1	0	1	$\bar{B}$	$(A+B)\ más\ (A\cdot\bar{B})$
0	1	1	0	$A\odot B$	$A\ más\ B$
0	1	1	1	$A+\bar{B}$	$A\ más\ (A+B)$
1	0	0	0	$\bar{A}\cdot B$	$(A+\bar{B})\ más\ 0$
1	0	0	1	$A\oplus B$	$A\ menos\ B\ menos\ 1$
1	0	1	0	B	$(A+B)\ más\ (A\cdot B)$
1	0	1	1	$A+B$	$A\ más\ (A+\bar{B})$

[31] Una ***UAL*** de uso corriente durante muchos años que facilitó enormemente el diseño digital fue la 74181 (circa 1960) en lógica ***TLL***.

1	1	0	0	$"0"\ lógico$	$menos\ 1\ (compl. a\ 2)$
1	1	0	1	$A \cdot \bar{B}$	$(A \cdot \bar{B})\ menos\ 1$
1	1	1	0	$A \cdot B$	$(A \cdot B)\ menos\ 1$
1	1	1	1	A	$A\ menos\ 1$

La unidad es relativamente rápida[32] y suma dos números en aproximadamente 6.5ns con un acarreo generado en menos de 5ns.

9.7.1 La Unidad Aritmética y Lógica hoy en Día

Los diseños de ***CPU*** basados en circuitos externos hoy en día no son comercialmente viables debido al comparativamente bajo precio y al alto rendimiento de los microprocesadores. Todo microprocesador moderno de alto rendimiento incluye en el propio chip la ***UAL*** como parte de su diseño. Un µcomputador actual puede incluir hasta 40 ***UAL*** en un mismo circuito junto con hasta 10 ***UPF*** (vea la siguiente sección).

9.8 La Unidad de Punto Flotante

La mayoría de nosotros no concebimos una respuesta distinta a un cálculo matemático realizado en un microprocesador distinto. Este caso era materia común hasta 1985. Se debía a que no existía un estándar que regulase los números de punto flotante. Considere por ejemplo el número π. Este número se debe representar en una cantidad finita de espacio siendo que es un número infinito. El estándar IEEE 754 cambió todo eso. Fue presentado en 1985 y se adoptó casi de inmediato.

Al igual que la ***UAL***. Los cálculos de punto flotante se realizaban en una unidad independiente de la ***UPC*** llamada Unidad de Punto Flotante ***UPF*** (Floating Point Unit; ***FPU***) o procesador de punto flotante que realizaba alrededor de 1 millón de operaciones de punto flotante por segundo o ***FLOPS*** (floating-point operations per second). Este procesador toma especial importancia para aquellas personas que usan la computadora principalmente para juegos. Los juegos de simulación y aquellos en los que se dibujan paisajes que cambian rápidamente representan texturas realistas, requieren de cálculos complejos de punto flotante que se delegan a circuitos especializados (frecuentemente tarjetas gráficas costosas que incluyen su propio microcomputador y memoria; ***GPU***).

Hacia la Luna

El cohete de lanzamiento de satélites Ariane 5 501 perdió el control justo después de su despegue en 1996 explotando contra el suelo junto con su carga de 500 mil millones de dólares. Se trazó el problema hasta un lenguaje de programación que hizo caso omiso del manejo de fallos de punto flotante recomendados en las especificaciones IEEE 754.

[32] Para su época, no para los estándares modernos.

Si los circuitos adecuados no se incluyen, queda la solución de proveer programas especializados que se encargan de realizar dichas subrutinas de punto flotante incluyendo las funciones trascendentales (trigonométricas, logarítmicas, diferenciales y otras).

Para una arquitectura, en particular de una microcomputadora, las rutinas de punto flotante pueden emularse con funciones de librerías de software que se incluyen como:

- Microcódigo en la propia ***UPC*** (concepto no muy difundido).
- Como una función del sistema operativo. Esto toma tiempo del procesador, pero evita el coste extra de circuitos externos o anexos.
- En el código del propio usuario.

Anteriormente los diseños incluían un socket extra junto con la lógica para que la ***UPC*** se comunicase con este tipo de procesador. Hoy en día y debido a la extrema miniaturización, no es extraño encontrar la ***UPF*** dentro del corazón mismo de la ***UPC*** o al menos, programas de emulación de matemática de punto flotante basados en aritmética de punto fijo que se ejecuta en una de las ***UAL*** interna basada en enteros, usando uno o varios registros especializados.

9.9 Resumen

Se analiza la Unidad Aritmética y Lógica (***UAL***) que es la parte de la Unidad de Procesamiento Central (***UPC***) que se encarga, como su nombre lo dice, de realizar las operaciones lógicas y aritméticas con los datos que la Unidad de Control obtiene de la memoria central o de los registros de trabajo de la ***UPC***. Forman parte importantísima de esta ***UAL*** los sumadores, restadores, multiplicadores y divisores de los que se dan ejemplos.

9.9.1 Puntos Importantes del Capítulo

- En la Unidad Aritmética y Lógica (***UAL***) se realizan las operaciones lógicas y aritméticas.
- La Unidad de Control dirige las operaciones de la ***UAL***.
- La Unidad de Memoria provee de los datos a la ***UAL***.
- En un registro especial llamado Acumulador, es donde la ***UAL*** guarda los datos de sus operaciones.

- El medio sumador suma dos bits binarios sin considerar acarreo de bits anteriores; el sumador completo considera este acarreo previo.
- Los sumadores paralelos y seriales son realizaciones prácticas de los circuitos de suma.
- Usando el complemento de un número nos evitamos la resta que puede entonces realizarse con la suma.
- En la multiplicación y división encuentran su aplicación los registros de corrimiento. Estas operaciones pueden realizarse con sumas y restas consecutivas.
- La Unidad de Punto Flotante o ***UPF***, se encarga de la matemática de punto flotante ya sea usando circuitos especializados o subrutinas de emulación.

10 La Memoria

10.1 Tipos de Memoria

La memoria de la computadora no está concentrada en un sólo lugar; los dispositivos de almacenaje están distribuidos en toda la máquina. En la parte más interna encontramos los registros de operación que son registros de biestables que se usan en la unidad de control y la unidad aritmética y lógica de la computadora. Los cálculos se realizan con los datos que se toman de estos registros, por ejemplo: la suma, multiplicación y corrimientos son todos realizados en estos almacenamientos provisionales. El proceso actual de información se realiza en la localidad de estos registros.

Viendo hacia afuera del ***CPU***, la siguiente categoría de registros de almacenamiento que encontramos es llamada ***memoria de alta velocidad***, ***memoria interna*** o ***memoria principal***. Esta sección de la memoria de la computadora consiste en un conjunto de registros de almacenamiento, cada uno de los cuales es identificado con una dirección (localización de cada registro de memoria) que permite a la unidad de control, ya sea escribir o leer, de un registro en particular. Para que un programa (conjunto de instrucciones) pueda ser ejecutado, es necesario que resida antes en esta memoria.

Es deseable que la velocidad de operación en esta sección de la memoria de la computadora sea tan rápida como sea posible, puesto que la mayoría de las transferencias de datos de y hacia la sección de procesamiento de la información de la máquina será a través de la memoria principal; por esta razón los dispositivos de almacenamiento con un tiempo de acceso (tiempo que transcurre entre el momento que se solicita la información y el momento en que el dispositivo tiene una respuesta) rápido son generalmente elegidos para la memoria principal.

Como regla general, los dispositivos o, no son lo suficientemente rápidos para realizar esta función satisfactoriamente o lo son, pero su precio es prohibitivo. La mayoría de ellos no poseen la capacidad de almacenamiento requerida. Como resultado, memoria adicional, llamada ***memoria auxiliar*** o ***memoria secundaria***, se agrega a la mayoría de las máquinas. Esta sección de la memoria de la computadora se caracteriza por un bajo costo por dígito

almacenado, pero generalmente tiene una velocidad de operación mucho menor que los registros de operación o que la memoria principal. Esta sección de la memoria es algunas veces designada como memoria de respaldo, pues su función es manejar cantidades de datos mucho mayores de las que pueden ser almacenadas en la memoria interna.

Los dispositivos más externos y últimos de almacenamiento son usados para introducir información a la computadora del "mundo externo" y para guardar los resultados de la computadora para el usuario de la misma. Este medio de almacenaje consiste generalmente en tarjetas perforadas (en desuso), cinta de papel perforada (en desuso), cintas magnéticas (en desuso), discos duros magnéticos (***HDD***; comienza su obsolescencia) o discos de estado sólido[33] (***SSD***) y las salidas de la máquina consisten, generalmente, en caracteres impresos ya sea en la pantalla o en papel.

Cada una de las divisiones de la memoria tiene ciertas características. Por ejemplo, la importancia de velocidad es muy alta en los registros de operación. Estos registros generalmente deben realizar operaciones varias veces mayor a la velocidad de la memoria principal. La memoria principal también requiere grandes velocidades de operación, pero, como es deseable almacenar grandes cantidades de datos (de 600,000 a 10^9 bits) en esta sección de la memoria, un compromiso entre el costo y la velocidad debe buscarse. El mismo tipo de compromiso se realiza en el caso de la memoria auxiliar en el que se almacenan entre 360,000 y 10^{15} bits, siendo muy caro utilizar los mismos dispositivos que para la memoria principal.

Es importante darse cuenta al considerar la velocidad de operación, que antes de que una palabra puede ser leída, es necesario localizarla. El tiempo para localizar y leer una palabra de memoria es llamado ***tiempo de acceso***. El procedimiento para localizar información puede ser dividido en dos clases:

- Acceso Aleatorio
- Acceso Secuencial

[33] Discos de estado sólido, SSD, inicialmente construidos de "burbujas magnéticas" demasiado caras y lentas. Su tecnología cambió a memoria tipo "Flash" (compuertas ***NOY*** y ***NOO***) más barata, que almacena datos no volátiles y memoria dinámica de acceso aleatorio (DRAM).

Los dispositivos de almacenaje que tiene acceso aleatorio son aquellos en los que la localización dentro del dispositivo puede ser seleccionada arbitrariamente sin seguir ningún orden y el tiempo de acceso aproximado es casi igual para cada una de las localidades de memoria del dispositivo. Un registro de biestables es un ejemplo de un dispositivo de almacenamiento de acceso aleatorio.

Por otro lado, los dispositivos de acceso secuencial necesitan ser recorridos en orden para llegar a cierta localidad de memoria, por lo que el tiempo de acceso varía de acuerdo con la localidad. Los dispositivos de acceso secuencial poseen a su vez dos divisiones:

- Dispositivos de almacenaje de acceso directo
- Dispositivos de acceso serial

Los primeros tienen direcciones definidas, pero su tiempo de acceso para alcanzar los datos en una dirección dada pueden variar; por ejemplo, el tiempo para localizar datos en una cabeza de discos movible depende de la posición de la cabeza y del disco cuando la dirección es dada. Los dispositivos de acceso serial son realmente seriales y secuenciales en sus propiedades de acceso; la cinta magnética (en desuso) es un ejemplo clásico donde para llegar a la localidad deseada es necesario pasar por todas las anteriores.

Otra forma de subdividir los dispositivos de almacenaje es de acuerdo con si son dispositivos del tipo

- Dinámicos o
- Estáticos

En los estáticos, la información, una vez grabada en su localidad, no es olvidada o borrada por el tiempo en que el dispositivo tenga energía eléctrica o la información no se cambie explícitamente. Como contraparte, en un dispositivo dinámico, una vez guardada la información en una localidad, ésta se olvida en un tiempo finito, por lo que se debe recordar o “refrescar” constantemente.

10.2 Memoria de Acceso Aleatorio

La memoria de una computadora se organiza en palabras de longitud física, cada una de ellas ocupa una localidad o dirección predeterminada dentro de la memoria. Como se puede observar en la figura *10.1* una memoria dada se divide generalmente en ***N***

palabras donde ***N*** es una potencia de 2. Cada palabra tiene el mismo número de bits llamados longitud de la palabra.

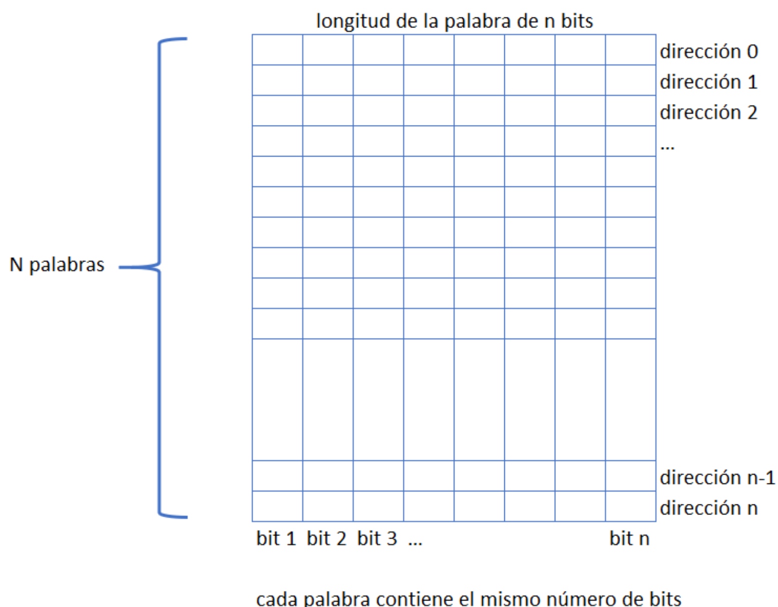


Figura 10.100 Organización de la memoria.

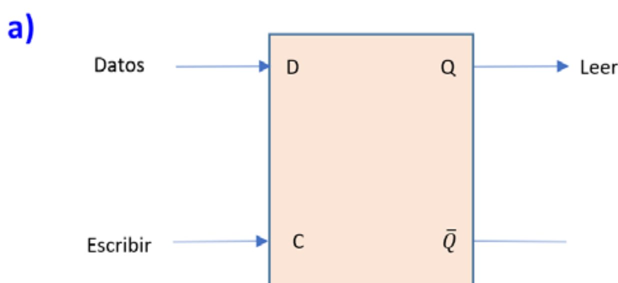
La dirección de los números en la memoria corre consecutivamente comenzando con la dirección cero hasta la máxima que la computadora puede direccionar. Es importante entender la diferencia entre contenido de la memoria y la dirección de la memoria. Una memoria es como un gran gabinete que contiene tantos cajones como hay direcciones en memoria. En cada cajón hay una palabra y la dirección de la palabra está escrita en la parte externa del cajón. No removemos la palabra en una dirección cuando la movemos, únicamente cambiamos el contenido de la dirección cuando almacenamos o escribimos una nueva palabra.

Desde el punto de vista externo, una memoria principal de alta velocidad es muy similar a una "caja negra" con un número de localidades o direcciones en las que los datos pueden ser almacenados o leídos. Cada dirección o localidad contiene un número fijo de bits binarios y a este número se le llama longitud de la palabra de la memoria. Una memoria que contenga 4,096 localidades distintas de memoria, cada una de ellas capaz de almacenar 16 bits binarios, es llamada "memoria de 16 bits 4,096 palabras" o en el lenguaje vernáculo de la computadora "memoria de 4K 16 bits". Puesto que las memorias generalmente vienen en potencias de 2, el sufijo K significa en este caso 1,024 y no 10^3, como en el caso decimal, pues se entiende que las 2^N localidades de memoria

están disponibles. Por ejemplo, una memora de 16 bits de 2^{16} palabras es llamada "memoria de 64K 16 bits".

Las memorias pueden ser leídas (los datos pueden ser tomados de ella) o escritas (los datos son almacenados en la memoria). Las memorias en las que es posible leer y escribir son llamadas memorias de ***lectura y escritura*** y erróneamente conocidas como ***RAM*** (Random Access Memory, memoria de acceso aleatorio) ya que las memorias de sólo lectura (ver siguiente párrafo) también entran en esta clasificación. A las memorias en las que solamente es posible leer y no permiten escritura, son llamadas memorias de ***sólo lectura*** o ***ROM*** (Read only memory).

En su forma más básica, la memoria de la computadora puede ser representada como una unidad de almacenamiento formada por un sólo flip−flop. En cualquier momento el flip−flop puede ser leído para determinar el estado en el que está y, también en cualquier momento, el estado del biestable puede cambiarse aplicando las señales adecuadas y el pulso de reloj correspondiente (ver figura *10.2a*).



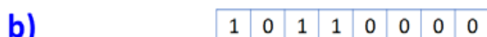


Figura 10.101 Flip-flop usado como elemento de memoria.

Una palabra de información binaria puede almacenarse en grupos de flip−flops, por ejemplo, 8 bits en 8 biestables. Las entradas y salidas de los flip−flops se encuentran interconectadas de manera que el grupo de bits binarios entre y salga de forma serial o secuencial (uno a la vez), o sean movidos en paralelo (todo el grupo a la vez).

Cuando se trata de registros dentro de la ***UPC*** (***CPU***) los programadores usualmente los representan como pequeñas cajas una junto a otra y, dentro de ellas, un cero o uno lógico que representa su contenido (ver figura *10.2b*).

10.2.1 Dirección

Incrementando el número de los flip−flops de almacenamiento podemos aumentar el número de bits disponibles. Pero un problema fundamental surge: ¿Cómo poder acceder a cada una de las localidades sin confusión?

La solución es usar una forma de dirección. Podemos aplicar un único grupo de dígitos binarios a una serie de líneas de selección y de esta forma una localidad única queda disponible. En la figura *10.3* mostramos una memoria de semiconductores estática típica arreglada de forma que sus biestables o “celdas” como algunas veces son llamadas, puedan ser seleccionadas independientemente con un código binario único llamado ***palabra de dirección***.

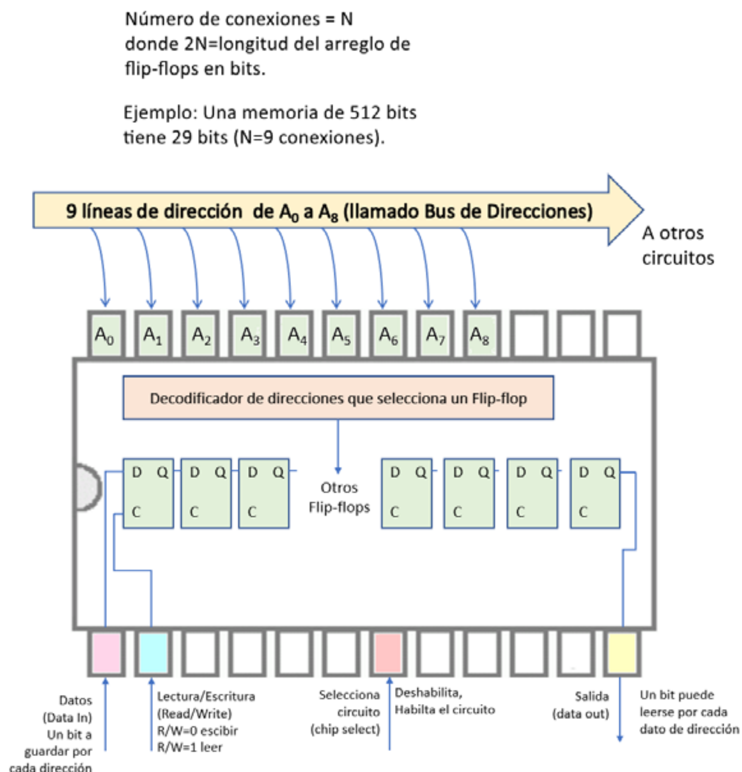


Figura 10.102 Organización de un circuito de memoria.

Una vez que la celda ha sido localizada, el puerto de entrada del ***CI*** es habilitado y un 0 o un 1 se puede escribir en la celda correspondiente. Dentro del circuito las celdas se organizan en una matriz de *x* por *y*, con la mitad de las líneas de dirección localizando las coordenadas de *y*. Pero por ahora es suficiente considerar a las

celdas como una larga cadena de biestables y la dirección como un selector de alguna forma de la celda correcta.

Para leer una localidad, es necesario:

1. Habilitar el circuito.
2. Tener disponible la dirección a la entrada del ***CI.***
3. Colocar la línea de lectura/escritura en 0 lógico.
4. Leer la información en la línea de salida.

Nótese que las líneas de dirección del circuito son igual a ***n*** donde 2^n es el número de biestables de almacenamiento. Por ejemplo, un circuito de memoria de 256 localidades se organiza de forma tal que sólo tiene una línea de entrada y una de salida y necesita exactamente 8 líneas de dirección llamándosele una "memoria de 256 x 1".

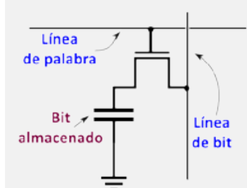
10.3 Memoria Dinámica

El circuito descrito en la sección anterior es del tipo estático (***SRAM***) donde la información grabada en cada celda no cambia con el tiempo. Este tipo de arreglo usa 3 transistores por cada localidad lo que da como resultado un área muy grande para grandes cantidades de celdas. Claro está que, conforme la tecnología evoluciona, los costos de fabricación y sus técnicas se mejoran, pero los diseñadores estaban buscando dispositivos que almacenaran más allá de los 16K de información.



Robert H. Dennard (1932-)

Ingeniero Eléctrico estadunidense que en 1968 patentó la memoria *DRAM* construida a base de transistores tipo *MOSFET*.



La solución al problema de la alta densidad de los circuitos vino de un nuevo tipo de dispositivo de almacenaje llamado ***RAM*** dinámica (***DRAM***). La celda de memoria dinámica es una forma ingeniosa de reducir el área del dispositivo de almacenamiento y, por lo tanto, aumentar su capacidad[34]. Consiste en un sólo transistor y un capacitor asociado; si el capacitor no está cargado, representa un 0 lógico y si el capacitor tiene carga, representa un 1 lógico. El transistor del tipo ***MOS*** sirve para amplificar la corriente de y hacia el capacitor. Desgraciadamente la carga del capacitor tiene fugas y eventualmente llega un punto en que ya no puede ser reconocido como un 1 lógico. Antes de que esto suceda, cada una de las celdas de la memoria debe leerse nuevamente y, si en

[34] Inútil dar cifras máximas pues la capacidad aumenta constantemente conforme la tecnología madura.

el capacitor existía un 1 lógico, éste debe renovarse; esta acción es conocida como ***refresco***.

Aún con todos los circuitos auxiliares necesarios para realizar el refresco de una memoria dinámica, ésta resulta ser más barata que la estática. Para realizar el refresco son necesarias dos líneas adicionales llamadas ***CAS*** (column address selection, selección de dirección de columna) y ***RAS*** (row address selection, selección de dirección de renglón). En la figura *10.4* representamos una memoria del tipo dinámico.

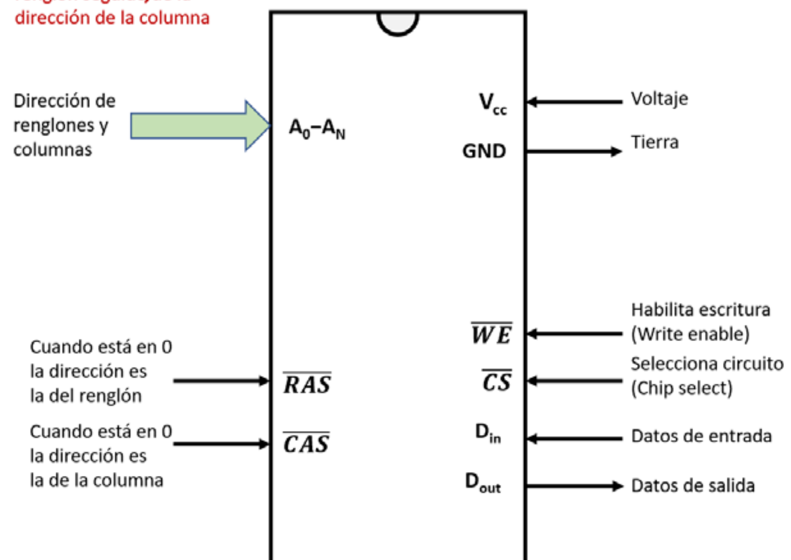


Figura 10.103 Memoria tipo RAM dinámica.

10.4 La Memoria de Sólo Lectura

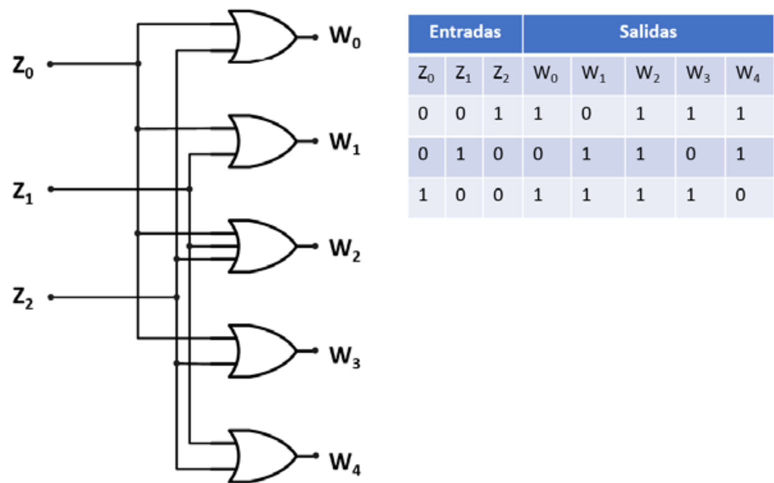
La memoria de sólo lectura (***ROM*** Read Only Memory) es un dispositivo de memoria utilizado para guardar información que no cambia. Hay una operación inicial en la que la información se graba en la memoria y posteriormente sólo es posible leer de ella y no sobrescribir la información. Generalmente la información es puesta en su lugar por el fabricante del dispositivo. Es deseable, sin embargo, en muchas ocasiones, poder grabar la información uno mismo en el dispositivo pues el costo de fabricar unos pocos ***ROMs*** es prohibitivo. Existen en el mercado una serie de circuitos que cumplen con esta característica y que son programables: las memorias de sólo lectura programables o ***PROM*** (Programmable Read Only Memory) memoria de sólo lectura programable. Estos circuitos permiten grabar la información una sola vez y de ahí en

adelante se comportan exactamente igual a un ***ROM***. Los circuitos de memoria de sólo lectura programables y borrables o ***EPROM*** (Erasable Programmable Read Only Memory) se utilizan si se realizan circuitos experimentales y son constantes los cambios de diseño de datos y programas en la etapa inicial o se requiere de la flexibilidad de poder borrar de vez en cuando el contenido de la memoria ***ROM*** y reprogramarla. Estos circuitos pueden ser programados, tal como los ***PROM*** pero tienen una ventana de cuarzo en la parte superior del empaque por el que se puede borrar la información usando una luz ultravioleta.

El atributo más importante de la memoria ***ROM*** es que la información grabada en ella no se borra si el voltaje que requiere el circuito para funcionar es interrumpido. Tales memorias son conocidas como ***no volátiles***. En contraste, casi todo tipo de memoria ***RAM*** es ***volátil***.

La memoria ***ROM*** tiene muchas aplicaciones en sistemas digitales. Puede ser usada, por ejemplo, para contener valores arbitrarios de una tabla de verdad. Cuando una tabla de verdad tiene muchas variables lógicas de entrada y salida y su realización requiere de un gran número de compuertas, puede ser económicamente substituida por una memoria ***ROM***. Se usa también en conversión de códigos para despliegue numérico. Tiene una amplia aplicación en cálculo numérico donde se requiere de muchas operaciones que pueden ser substituidas por una búsqueda en tablas, tales como las funciones trascendentales, multiplicación, división, logaritmos, etc.

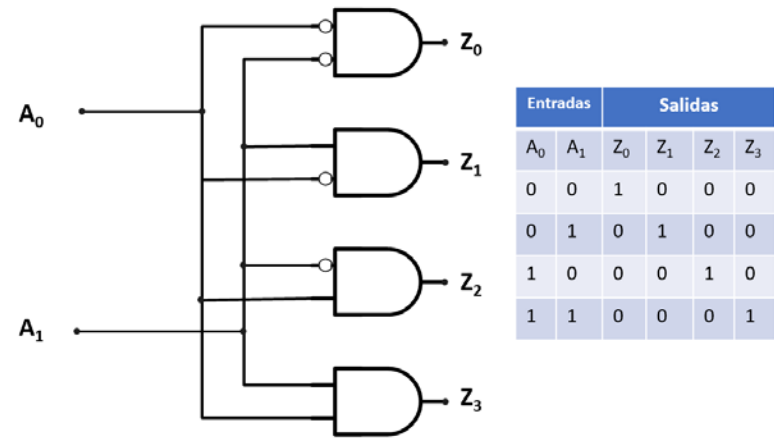
Al circuito ***ROM*** se le conoce como encodificador pues, de sus varias líneas de entrada, sólo una de ellas puede estar en cierto momento en uno mientras que las demás se encuentran en cero lógico. A su salida obtenemos un código arbitrario definido de antemano (caso exactamente contrario al de un decodificador). El encodificador se ve en esta aplicación como una memoria en la que se accede a su enésima localidad poniendo un 1 lógico en una y sólo una de sus entradas. El ***ROM*** responde colocando la palabra almacenada en esa localidad a su salida. En la figura *10.5* mostramos un ejemplo de esta aplicación en una memoria ***ROM*** de 3 palabras cada una de 5 bits.



Entradas			Salidas				
Z_0	Z_1	Z_2	W_0	W_1	W_2	W_3	W_4
0	0	1	1	0	1	1	1
0	1	0	0	1	1	0	1
1	0	0	1	1	1	1	0

Figura 10.104 Encodificador realizado con compuertas tipo O.

Por lo general, en los sistemas digitales, la dirección de una palabra almacenada es una palabra codificada en binario. Para estos casos es necesario interponer un decodificador entre la dirección y la memoria ***ROM***. Mostramos un ejemplo de decodificador de 2 a 4 (en la simbología estándar de los decodificadores se hace relación a sus líneas de entrada con respecto a las de salida) en la figura *10.6*.



Entradas		Salidas			
A_0	A_1	Z_0	Z_1	Z_2	Z_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Figura 10.105 Encodificador realizado con compuertas tipo Y.

Nótese que si el circuito de la figura *10.6* se usa como interfaz para el ***ROM*** de la figura *10.5*, sólo 3 de las 4 líneas posibles son utilizadas.

Como una conveniencia al usuario, los fabricantes de ***ROMs*** proveen generalmente un decodificador integrado en el circuito (ver figura *10.7*). Tales circuitos son conocidos con el nombre de ***ROM*** codificado. Al proveer un decodificador en el mismo circuito, los

fabricantes se hacen un servicio a ellos mismos, pues aumentar el número de patillas de interconexión en un circuito integrado es costoso y en un ***ROM*** de *M* palabras necesitaríamos ***M*** conexiones externas mientras que con el uso de un decodificador sólo se necesitan ***N*** entradas donde $2^N=M$.

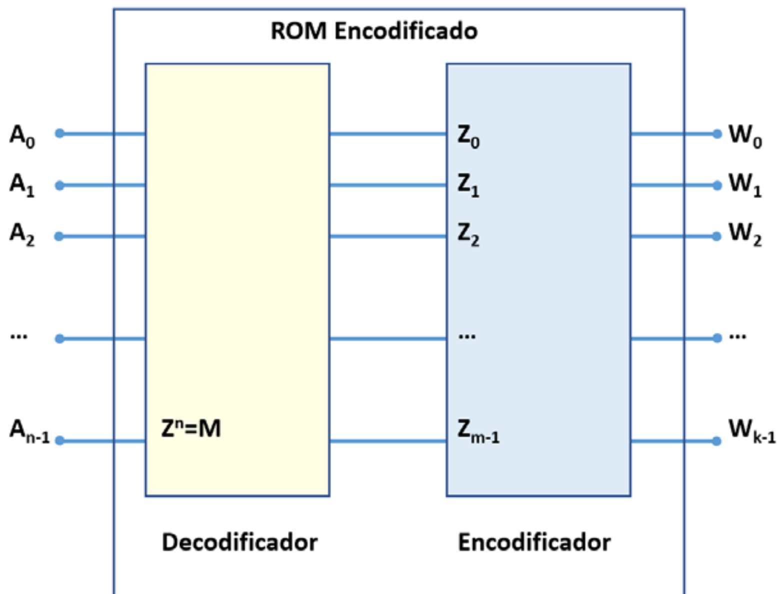


Figura 10.106 Memoria ROM con decodificador incluido.

10.4.1 Realización de una Memoria de Sólo Lectura

Una posible realización de una memoria de sólo lectura se muestra en la figura *10.8*. La estructura mostrada corresponde exactamente a una estructura de compuertas del tipo ***O***. El uso de diodos (figura *10.8a*) tiene sus desventajas pues la corriente debe ser proporcionada en su totalidad por el circuito externo. Se resuelve esta dificultad con el empleo de transistores bipolares en lugar de diodos en cada unión. Vea la figura *10.8b* en la que hemos mostrado sólo las conexiones de la línea W_1 para simplificar el diagrama. Observamos que todos los transistores conectados a la línea Z_1 tienen todos sus colectores en común con V_{CC} mientras que todas sus bases son comunes a la línea Z_1 por lo que a veces se prefiere substituir todos los transistores por uno sólo con emisores múltiples (como en el caso del transistor de entrada de la familia ***TLL***).

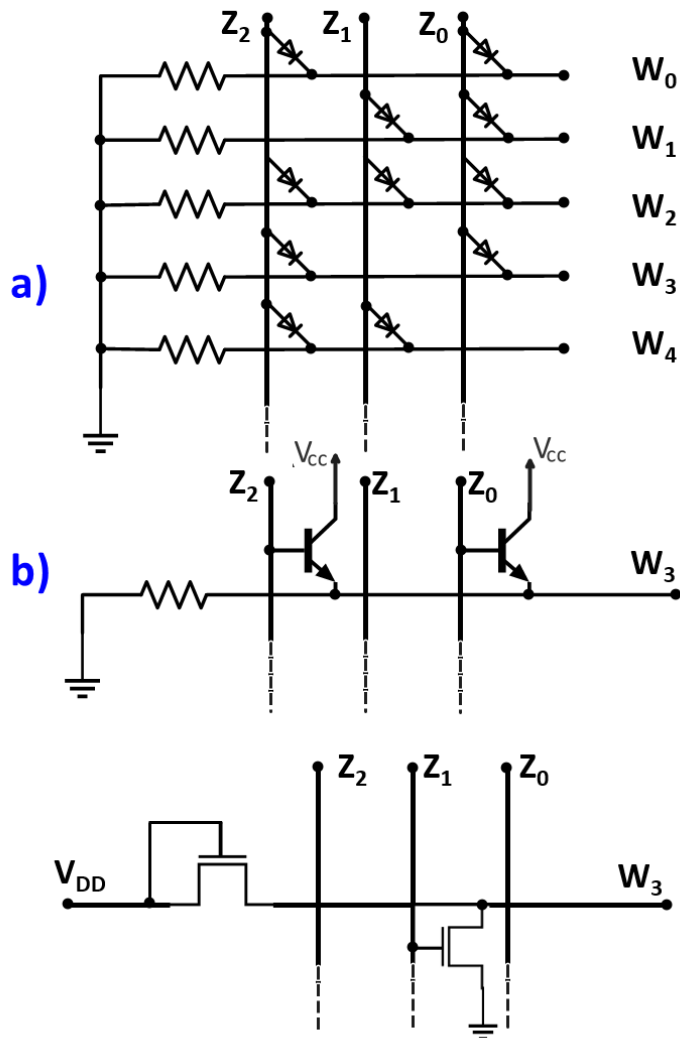


Figura 10.107 Distintos tipos de memoria ROM.

Se le nombra ***tiempo de acceso*** de la memoria al lapso que pasa entre el tiempo en que la dirección está disponible en las entradas del ***ROM*** y el tiempo en que éste responde teniendo disponible la salida. El tiempo de acceso para las memorias ***ROM*** fabricadas con tecnología bipolar puede ser tan baja como de 15ns (y menores).

El número de bits de una memoria es definido como el producto del número de palabras por el número de bits por palabra. Cuando el número de bits es grande (más de 1000) se prefiere la fabricación con dispositivos del tipo ***MOS*** (ver figura *10.8c*) que son más baratos y ocupan menos área. Los tiempos de acceso de una memoria ***ROM*** fabricada con la tecnología ***MOS*** son de alrededor de 60ns y se mejoran constantemente.

10.4.2 Memorias Programables y Borrables

Como se ve en la figura *10.8*, la memoria ***ROM*** forma una rejilla de líneas formada por las líneas de dirección y las de salida. Cada intersección entre líneas tiene un dispositivo (diodo o transistor) si su salida es un 1 lógico o, si no tiene dispositivo, su salida es 0 lógico. En una memoria ***ROM*** programable el fabricante coloca en cada intersección un dispositivo con un fusible. El usuario funde este fusible si no quiere conexión haciendo pasar por la línea un voltaje alto por un tiempo especificado por el fabricante. De esta forma el usuario puede programar el ***PROM*** quemando los fusibles de los dispositivos que no quiere utilizar.

En las memorias ***PROM*** borrables o ***EPROM***, el dispositivo en algunos casos se forma con un transistor del tipo ***PMOS*** de compuerta flotante. En estos transistores la compuerta se encuentra aislada de toda parte del circuito en su operación a voltajes normales. Resulta que si pasamos por las otras dos conexiones del transistor (fuente y drenaje) un voltaje alto, es posible establecer una carga negativa en la compuerta. La carga negativa dejada en la compuerta por este tipo de tratamiento deja al transistor correspondiente con un canal que ahora conduce. El ***EPROM*** puede borrarse al exponerse a la luz ultravioleta que sirve para descargar toda compuerta cargada negativamente de cada uno de los transistores ***MOS***.

10.5 Organización de la Memoria

Los distintos tipos de memorias vistos vienen en circuitos integrados que tienen, tanto distintos números de conexiones cada uno, como distinta organización interna y capacidad.

Cuando más de un tipo de memoria es usado en la memoria principal, debemos colocar a cada tipo de memoria en el llamado ***mapa de memoria***. Un mapa de memoria es un plan de dirección para todos los bits de las líneas de dirección.

Al conjunto de las líneas de dirección de un computador se le denomina genéricamente con la palabra en inglés de ***bus*** y se le conoce como bus de dirección. El bus no es más que un conjunto de líneas conductoras (alambres) que se especializan en una función y ésta es llevar las señales de un grupo determinado de circuitos a otros; así, existe el bus de datos, el bus de direcciones y otros que veremos en detalle conforme el libro avance.

Existen dos técnicas básicas para realizar la selección de los ***CI*** que necesitamos acceder en dado momento:

- Selección Lineal
- Dirección completamente decodificada

En la selección lineal se escoge dividir el número total de líneas de dirección entre los distintos circuitos. Por ejemplo, si escogemos la línea más significativa para la selección de un circuito de ***ROM***, el ***ROM*** es seleccionado cada vez que esta línea se encuentra en 1. Podemos entonces planear que nuestra memoria ***RAM*** se seleccione cuando la línea más significativa sea cero.

La ventaja esencial de la selección lineal es su sencillez: no es necesaria lógica complicada ni especial para realizar el mapa de memoria. Cada nuevo circuito que coloquemos es seleccionado por una línea dedicada (única y exclusiva) y es la forma más utilizada en pequeños sistemas de microcomputadoras.

Sin embargo, la selección lineal nos deja con sólo la mitad de las posibilidades cada vez que una línea independiente es seleccionada por lo que muchas veces es preferido el segundo método se acceso a la memoria: la dirección completamente decodificada.

Por otro lado, el objetivo de la dirección completamente decodificada es tener disponible el total de los bits direccionables sin utilizar líneas dedicadas para cada una de las secciones del mapa de memoria.

Para diseñar este tipo de mapa, es necesario dividir la memoria en bancos o páginas, donde cada página es seleccionada por medio de la salida de un decodificador según el número de páginas de memoria que tengamos.

Ejercicio

10.1 Diseñe una memoria con 3 circuitos ***ROM*** de 16K x 1 bit y 3 circuitos ***RAM*** de 256K x 1 bit utilizando selección lineal. Haga un diagrama completo del mapa de memoria resultante.

Como ejemplo, diseñemos el mapa completo de memoria de una computadora con 16 líneas de dirección (64K bytes)

longitud de palabra de 8 bits usando dos bancos de dispositivos ***RAM*** de 16K x 1 bits y 1 dispositivo ***ROM*** de 16K x 8 bits.

Comenzamos por dividir el mapa de memoria entre los dispositivos que tenemos disponibles y podemos ver, en este caso, que es posible dividir perfectamente en 3 áreas de memoria, pero como para controlar 3 líneas es necesario 2 bits, será mejor dividir en 4 áreas una de las cuales no tiene uso (ver la figura *10.9*).

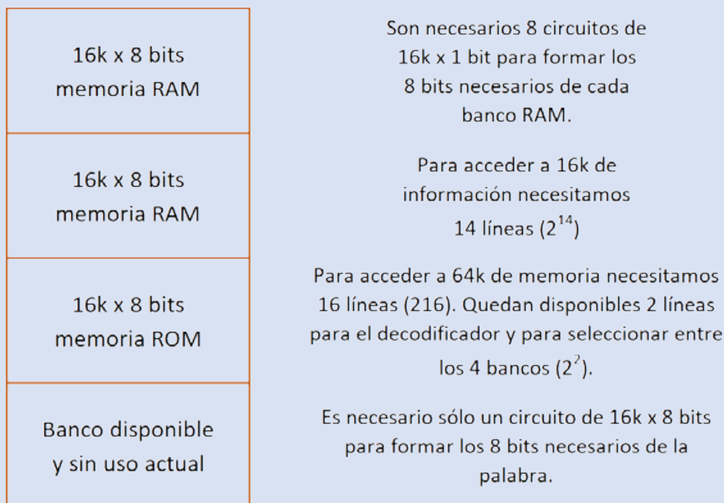


Figura 10.108 Mapa de memoria.

Escogemos las 2 líneas más significativas de dirección (A_{14} y A_{15}) para alimentar nuestro decodificador de 2 a 4 (2^2=4). El resultado final se muestra en la figura *10.10*.

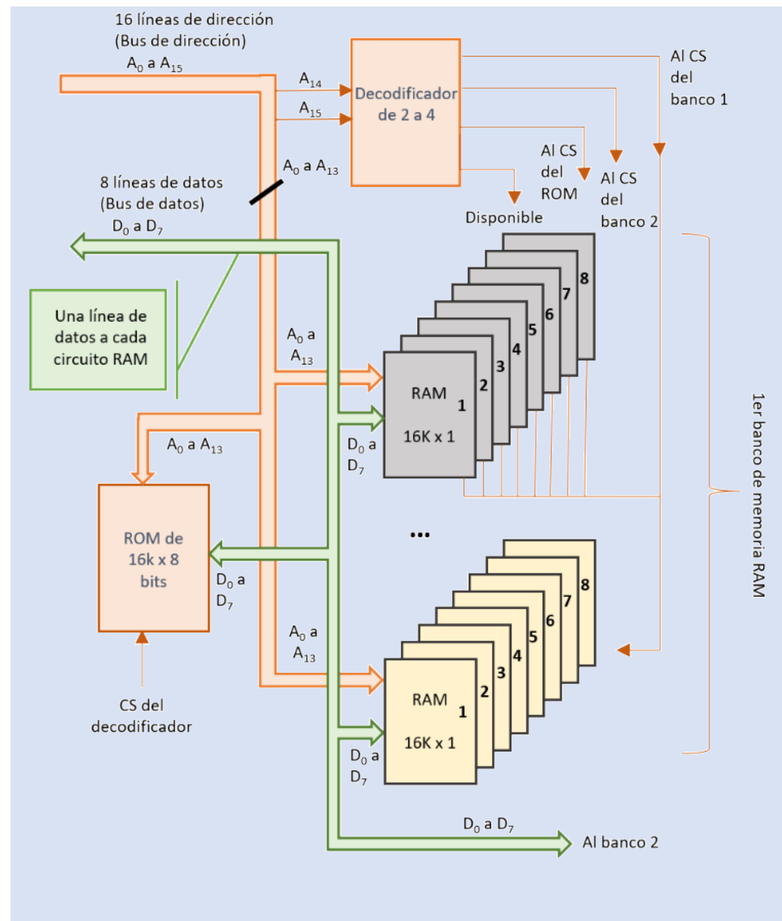


Figura 10.109 Realización de un mapa de memoria.

10.6 Memoria Cache (Antememoria)

Debido al constante incremento de la velocidad de las ***UPC***, las memorias principales no han podido mantenerse a la altura. Es inútil contar con la última generación de procesador si no podemos hacer que "corra" a su máxima velocidad si el ***CPU*** debe esperar constantemente a su "pareja", la memoria, que no le proporciona datos. Su desempeño sufre enormemente. La respuesta surge en el uso de lo que se denomina memoria ***cache*** o ***antememoria***. El cache es una memoria intermedia (búfer del inglés buffer) sumamente rápida, aunque cara, que trae los datos que el procesador requiere para que pueda trabajar con cierta holgura. Pero esta memoria cache requiere una característica extra: inteligencia.

Un búfer es una memoria aleatoria tipo ***FIFO***[35] o ***FILO***[36], pero una memoria cache es un búfer de alta velocidad que trabaja en estrecha colaboración con el ***CPU*** trayendo los datos que se requieren de la memoria principal para que estén a la mano cuando se requieran.

Con el avance en la monitorización y la disminución de coste de la producción en masa, las memorias se han vuelto aún más rápidas, densas y baratas. Con el tiempo ha surgido una nueva generación de memoria extremadamente rápida capaz de responder a ciclos de reloj extremadamente altos.

La memoria de acceso aleatorio estática o ***SRAM*** que, a diferencia de la ***DRAM***, no requiere refrescar su contenido, es una memoria de coste muy alto, pero extremadamente rápida.

La memoria cache se forma precisamente de la memoria ***SRAM*** que en un principio se colocaba cerca del ***CPU*** en un circuito aparte y hoy en día, debido al bajo coste de producción, se acostumbra integrar dentro del mismo circuito del procesador.

Una analogía que podemos usar para la memoria cache sería la siguiente: Supongamos que siempre desayuna a las 8:00 AM antes de irse al trabajo y cuenta con la ventaja de una cocinera en casa. Usted es el ***CPU***, la cocinera sería el controlador del cache, la cocina donde se prepara el desayuno correspondería a la memoria principal.

Al llegar al comedor la cocinera le pregunta lo que se le antoja. Usted pide, como es su costumbre, un jugo de naranja fresco. La cocinera desaparece y regresa 5 min después con un jugo de naranja que usted bebe en 2 min. Llama a la cocinera y, ya en la desesperación le pide un pan tostado con mantequilla y mermelada de fresa además de un café. Diez minutos transcurren hasta que la cocinera aparece con la orden. Usted tarda sólo cuatro minutos en consumir todo (se hace ya tarde). ¿Algo más?

La experiencia consistió en largas esperas a que todo se preparara seguido de una actividad frenética de consumo del alimento. La cocina y la cocinera no pudieron seguirle el ritmo. Pero la cocinera y el cocinero no son tontos, notaron su enfado y molestia y dado que todos los días desayuna lo mismo al tercer día lo sorprenden

[35] Primero que entra primero que sale.
[36] Primero que entra último que sale.

y cuando llega a la mesa a las 8:00 en punto ya le tienen su jugo de naranja y sus tostadas con café. ¡No hay espera! Se le agregó inteligencia al controlador del cache para predecir lo que desea que se traiga de memoria y en este caso hubo un ***acierto del cache*** (cache hit). La experiencia es satisfactoria: 0% de espera y 100% de proceso.

Es sábado, llega usted a la cocina y ya está ahí su jugo de naranja y tostadas con café. ¡Un momento! Yo no pedí tostadas. Los sábados yo desayuno jugo y huevos. Hubo un ***fallo del cache*** (cache miss); habrá que traer nuevos datos de la memoria principal desechando casi todo.

Para mitigar la dramática ralentización cada vez que el cache de 1er nivel (el más cercano al procesador) falla, se introduce el cache de 2do nivel. Si continuamos con la analogía de la cocinera podemos suponer que tendría platillos de repuesto ya preparados en la entrada de la cocina por lo que sólo tendría que llevarse lo que usted no quiere y regresar poco tiempo después con el nuevo pedido. Hoy en día este cache de 2do nivel también está incluido en el circuito integrado del ***CPU***.

Si esto fuese poco, algunos diseños de ***CPUs*** también incluyen un 3er nivel de cache en el propio circuito del procesador tratando de que los accesos a la, relativamente lenta memoria secundaria, queden limitados.

En la figura *10.11* mostramos dos diagramas que ayudan a comprender las distintas memorias y su organización.

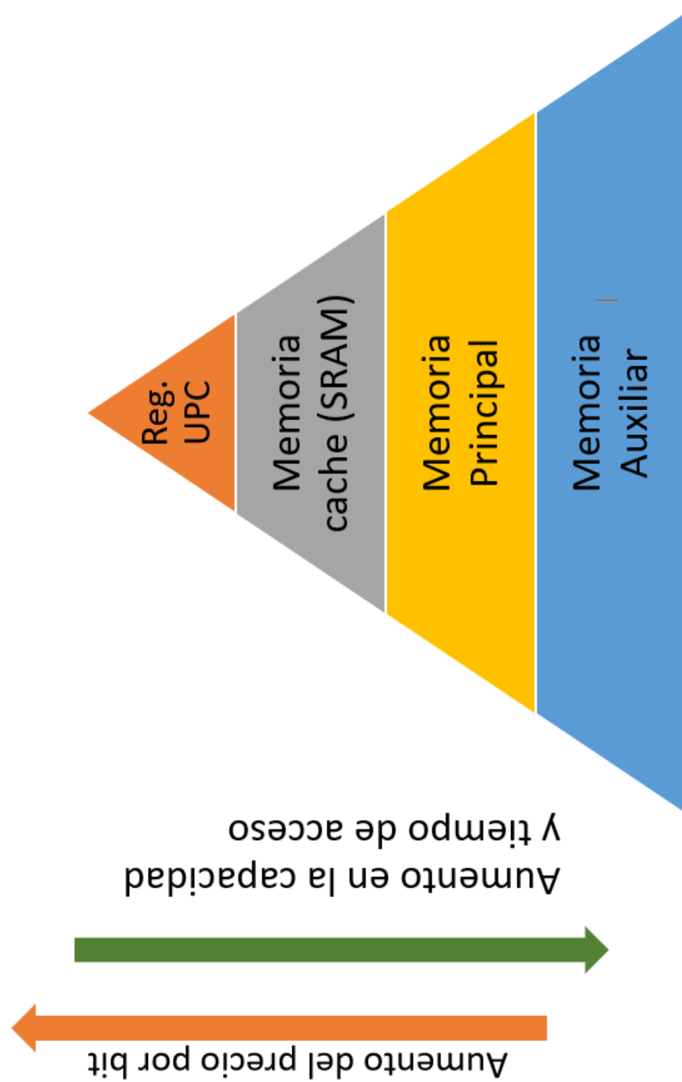
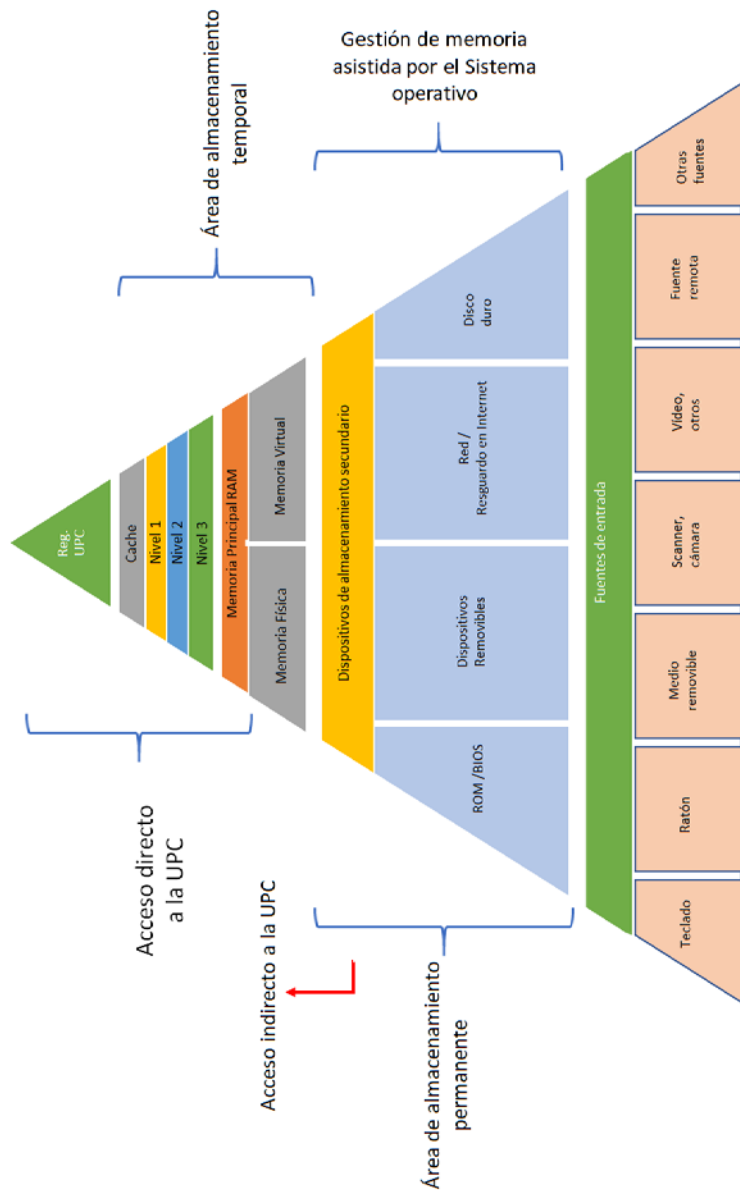


Figura 10.110 Diagramas de memoria.



10.7 Resumen

La memoria es la parte que hace que la Unidad de Proceso Central pueda funcionar. Si contamos con memoria y una ***UPC*** es más que suficiente para realizar un pequeño sistema; obviamente es necesaria toda una serie de dispositivos para introducir y obtener la información (analizada en el capítulo *15*). La memoria es clasificada en varios tipos y se estudia principalmente la memoria principal ***RAM*** y ***ROM***. En el capítulo *15* se analizará la memoria secundaria o auxiliar.

Es importante entender el principio de organización de la memoria para poder entender el sistema como un todo, pues la información posterior de la arquitectura del sistema y su clasificación depende de esta comprensión.

10.7.1 Puntos Importantes del Capítulo

- La memoria es dividida en Registros, Memoria Principal y Memoria Secundaria.
- El tiempo de acceso es el tiempo de respuesta de una memoria.
- El proceso para localizar información se divide en aleatorio y secuencial.
- Los dispositivos secuenciales pueden ser de acceso directo o de acceso serial.
- Hay dispositivos de memoria estáticos y dinámicos.
- Los dispositivos de acceso aleatorio se dividen genéricamente en ***RAM*** (lectura y escritura) y ***ROM*** (de sólo lectura).
- La dirección es la localidad de memoria donde se encuentra una información.
- La dirección es generada por la ***UPC***.
- La memoria dinámica debe ser "refrescada" continuamente para no perder su información.
- La memoria de sólo lectura o ***ROM*** se conoce como encodificadora.
- La memoria ***ROM*** tiene las variedades ***EPROM*** y ***PROM*** que son programables una o muchas veces para luego convertirse en memorias de sólo lectura.
- El bus son conexiones conductoras (alambres) que comunican a los distintos dispositivos.
- La memoria es organizada en un mapa de memoria y, físicamente, se organiza por su forma de acceso en lineal o decodificada.
- La memoria cache es una memoria búfer situada entre la memoria principal y el ***CPU***. Cuenta con cierta "inteligencia" y se utiliza para que la ***UPC*** tenga rápidamente a la mano una parte de los datos de la memoria principal (secundaria).

11 La Unidad de Procesamiento Central

Toda computadora tiene una unidad de procesamiento central o ***UPC*** (Central Processing Unit o ***CPU***), que forma el "cerebro" de la máquina computadora. La ***UPC*** está formada por diferentes partes interconectadas entre sí de forma tal que permite ciertas operaciones lógicas. En los albores de las computadoras modernas (1950-60) se requería un chasis sumamente amplio para albergar los componentes de la ***UPC***. Varias tarjetas bastante grandes que contenían cientos de circuitos y eran interconectadas por cables del tamaño de una manguera de bombero no eran una vista extraña. Hoy en día, la ***UPC*** está contenida en unos pocos circuitos de alta integración o, en el caso de las microcomputadoras, en uno solo. La microcomputadora, junto con unos cuantos circuitos externos, forman una computadora funcional. Casi toda computadora, desde la más humilde hasta la más potente, con pocas excepciones, se forman de una o varias microcomputadoras interconectadas.

Los componentes del sistema de computación deben interconectarse de alguna forma. La manera en que se realicen las conexiones y cómo se haga la comunicación afecta las características de desempeño del sistema profundamente.

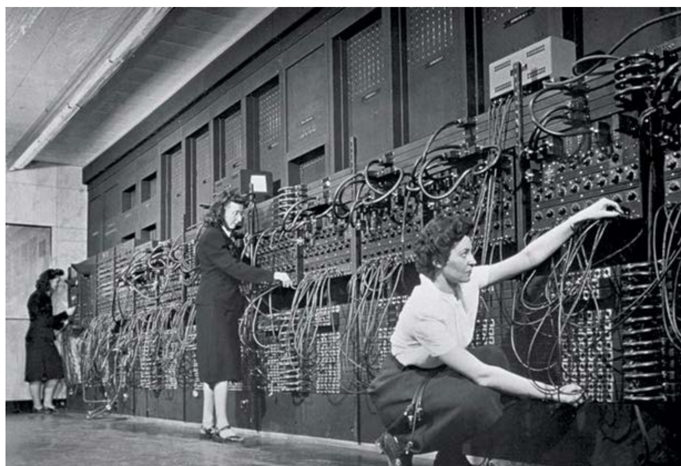


Figura 11.111 Programando la ENIAC, de las 1eras computadoras. Circa 1946.

En las primeras computadoras y en algunas de las actuales, la ***UPC*** se interconectaba con cada uno de los dispositivos externos a la ***UPC*** por medio de un cable separado. Este método tiene la desventaja de que muchos cables (llamados buses) distintos deben

controlarse y se requiere de una cantidad de circuitos considerables.

Para realizar la interconexión de forma más económica y para estandarizar la lógica de interfaz necesaria, una técnica muy popular consiste en interconectar a todos los sistemas externos a la ***UPC*** por medio de un sólo juego de cables o bus de interconexión. El bus consiste en los cables necesarios para encontrar la dirección de los componentes, controlarlos y transferir datos de y hacia ellos.

11.1 Las Microarquitecturas

Micro Arquitectura

En ingeniería de computación, la microarquitectura (a veces abreviada como µarquitectura), también conocida como organización de la computadora, es la forma en que una arquitectura del conjunto de instrucciones (*ISA*, Instruction Set Architecture) se implementa en un procesador.

Las arquitecturas ***ARM*** (Advanced *RISC* Machines), ***MIPS*** (Microprocessor without Interlocked Pipelined Stages) y *x86* son las familias de procesadores más comunes hoy en día.

Durante años, las ***ARM/MIPS*** han estado en el centro de los microprocesadores modernos y el diseño integrado. Con un enfoque en el bajo consumo de energía y un conjunto de instrucciones simple, los dispositivos móviles en particular se han beneficiado enormemente de este diseño del procesador.

Millones de procesadores orientados a entornos de alto y muy alto rendimiento, incluyendo computadores de escritorio, portátiles, servidores e incluso superordenadores, usan una u otra arquitectura indistintamente, predominando hasta ahora la x86 (lidereados por el fabricante de circuitos integrados Intel).

Los procesadores ***ARM/MIPS*** encajan en una familia llamada ***RISC*** (Reduced Instruction Set Computing; Computo con Conjunto Reducido de Instrucciones) que se centran en mantener un número de instrucciones básico y lo más simple posible. Las instrucciones simples tienen varias ventajas tanto para los ingenieros de hardware como para los de software. Dado que las instrucciones son simples, los circuitos necesarios requieren menos transistores, lo que resulta en más espacio de chip y/o chips más pequeños. Debido a esto, los procesadores ***ARM/MIPS*** normalmente integrarán muchos periféricos, incluidas las unidades de procesamiento gráfico o ***GPU*** (Graphics Processor Unit; Unidad de Procesamiento Gráfico).

Pero, usar sólo instrucciones simples, tiene un costo. Se requieren más instrucciones para realizar una tarea dada, dando por resultado un aumento en el consumo de memoria y tiempos de

ejecución más largos. Sin embargo, el procesador ***ARM/MIPS*** compensa esto con mayor velocidad del reloj y una cola de instrucciones más grande.

Los procesadores x86, por otro lado, encajan en una familia llamada *CISC* (Complex Instruction Set Computing; Computo con Conjunto Complejo de Instrucciones). A diferencia de los ordenadores *RISC*, las instrucciones disponibles en un *CISC* se centran más en realizar tareas complejas con mayor flexibilidad.

Por ejemplo, muchas máquinas basadas en arquitectura *RISC* realizan operaciones entre registros, lo que normalmente requiere que el programa cargue variables en los registros antes de realizar una operación. Una máquina basada en *CISC*, sin embargo, puede (o debe) ser capaz de realizar operaciones entre registros, entre un registro y una ubicación de memoria; incluso entre ubicaciones de memoria. Otras operaciones comunes incluyen multiplicación con números de punto flotante, corrimiento de números, instrucción para bucles, manipulaciones complejas de memoria, búsquedas de memoria y muchas otras.

Los diseños ***ARM/MIPS*** sobresalen por sus diseños de baja potencia que no requieren disipadores de calor. Su consumo es sumamente bajo, aun incluyendo una ***GPU***, periféricos y memoria. Esto sólo es posible debido a que usa menos transistores y velocidades relativamente más bajas (en comparación con las ***CPU*** de escritorio). Claro está que esto afecta al rendimiento del sistema y, por lo tanto, las operaciones más complejas tardarán más tiempo. Los procesadores *x86* consumen más energía que los ***ARM/MIPS*** debido a su complejidad.

En cuanto a la disponibilidad de software, los dispositivos basados en ***ARM/MIPS*** tienen la ventaja de ejecutar sistemas operativos diseñados para móviles mientras que los basados en *x86* tienen la ventaja de ejecutar casi cualquier sistema operativo que se pueda ejecutar en una PC de escritorio estándar.

En general, toda arquitectura tiene hoy en día una amplia gama de controladores y periféricos integrados y cada una encaja en su propio mercado claramente diferenciado.

En este libro hemos decidido apegarnos a la microarquitectura *x86* que tiene una aplicación inmediata para el lector lambda pero

que, con un poco de esfuerzo de su parte, puede aplicarse a cualquiera de las otras dos.

11.2 Microarquitectura x86

En la figura *11.2* vemos el esquema de una unidad de procesamiento central. Ésta se forma por la unidad aritmética y lógica encargada de las operaciones lógicas y aritméticas, la unidad de control encargada de la coordinación general, una serie de registros donde se almacena información requerida por la ***UPC*** de forma provisional y un bus de datos y control que interconecta a todas las unidades.

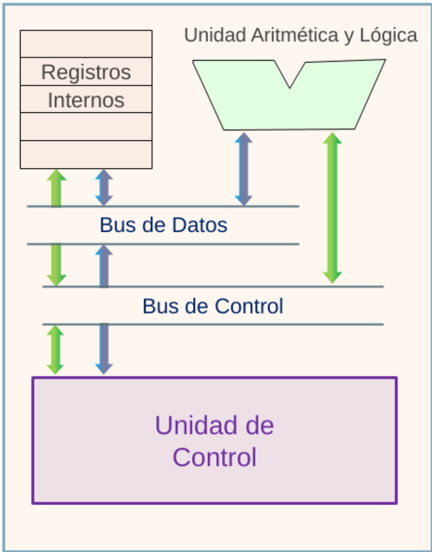


Figura 11.112 La Unidad de Procesamiento Central.

> Programa
>
> Conjunto de instrucciones que siguen una secuencia lógica para un fin determinado. El 1er programa informático se atribuye a Ada Lovelace, hija de Lord Byron.

La función principal de la ***UPC*** será la de interpretar la información que reside en la memoria principal como instrucciones o datos y actuar de acuerdo con un plan determinado por el código de la instrucción, generando así las operaciones necesarias de acuerdo con ese plan.

> Ningún programa puede ser ejecutado por la ***UPC*** si no reside antes, en forma parcial o total, en la memoria principal.

11.3 Interpretando el Contenido de la Memoria

Los datos almacenados en la memoria pueden ser interpretados como:

1. Datos binarios puros
2. Datos codificados en binario
3. Un código de carácter
4. Un código de instrucción

Estas cuatro categorías de interpretar datos pueden dividirse en dos más genéricas: datos e instrucciones.

Los datos binarios puros, los datos codificados en binario y los caracteres codificados tiene una cosa en común: son datos. El contenido de las palabras almacenadas en memoria puede ser usado por sí solo para operaciones o combinado con otras palabras de memoria, en formas dirigidas por un programa.

11.3.1 Datos binarios puros

Consideremos primero a los datos binarios puros que no están sujetos a ninguna interpretación. Es importante resaltar que podemos representar este tipo de datos en papel como un número binario, octal o hexadecimal (por sencillez) sin que tenga ningún impacto en el dato de la palabra. Por ejemplo, una palabra arbitraria de 16 bits, en la que es posible representar 2^{16} combinaciones, puede ser:

$1101\ 1011\ 1000\ 1011_2=155613_8=DB8B_{16}$

Se puede observar que la anotación en hexadecimal, además de ser más compacta, es mucho más fácil de apuntar, recordar y evita confusiones, por lo que la base 16 es la preferida entre la gente dedicada al diseño lógico o programación a nivel máquina.

11.3.2 Datos Codificados en Binario

El contenido de una palabra de memoria, interpretada como datos binarios puros, puede existir por sí solo o ser válido sólo cuando se combina con otras palabras para formar un código o un número mayor que sea utilizado con fines prácticos. No hay razón por la cual un dato binario no pueda formarse por ejemplo con 2 o más palabras, más aún, las palabras pueden no ocupar localidades continuas de memoria, aunque se prefiere por fines prácticos que así sea.

La palabra puede también estar codificada de acuerdo con un código establecido de antemano como puede ser exceso 3, ***BCD***, números con signo, etc. (todos ellos analizados en el capítulo 3).

11.3.3 Códigos de Caracteres

Una computadora no sería muy práctica si tuviésemos que meter cada uno de los datos como largas secuencias de unos y ceros o si las respuestas consistieran en números o letras en una representación binaria. Para la computadora debe ser posible manejar todo tipo de texto y otra información no numérica.

Para que el manejo de texto sea completo debe incluir al menos las 26 letras en minúsculas y mayúsculas; los números y una serie de signos especiales que usamos diariamente en nuestra comunicación como son el punto, la coma, etc. En conjunto, los caracteres más usados suman alrededor de 87[37]. Una palabra de 7 bits es suficiente para su representación, pero es universalmente aceptada la codificación utilizando 8 bits (ver capítulo 3).

En algunos sistemas con longitud de palabra de 8 bits, se usa el código ***ASCII*** de 7 bits y el bit libre se utiliza para verificar errores en los datos por medio de un procedimiento llamado ***paridad***. En otros sistemas se usa un código de 8 bits ***ASCII*** y se agrega a la memoria un bit más que sirve para la comprobación de paridad. A este bit se le conoce como bit de paridad.

La paridad consiste en una verificación sencilla de errores en la que se agrega un bit al código. La paridad puede ser:

- Par
- Impar

El bit de paridad se cambia a 1 o a 0 para que el número de bits 1 de una palabra sea siempre par o impar, según el tipo de paridad que se haya escogido. Si el tipo de paridad que se escoge es par, el bit de paridad se limpia o se fija para que el número total de unos de la palabra sea siempre par. Para clarificar el punto incluimos un sencillo ejemplo:

---

[37] Para muchos de los alfabetos occidentales. Recuerde que existen otros en los que 128 (2^7) símbolos no son suficientes.

Ejemplo
11.1 Proteja la palabra 101 0101 con un bit de paridad par.

Respuesta
Como el número de unos de la palabra ya es par, el bit de paridad (en color rojo) será en este caso 0 y el resultado final es:

0101 0101

Muchos esquemas muy elaborados se usan para verificar la consistencia de los datos y asegurarse que no contengan errores; aún más, existen algoritmos para determinar dónde están los errores y tratar de corregirlos. Estas técnicas de detección de errores no tienen que ver nada en particular con la arquitectura de las computadoras y, por lo tanto, no se discuten aquí. Para más información consulte la bibliografía, en especial el libro *Telecomunicaciones y Teleproceso*.

11.3.4 Código de Instrucciones

Las palabras de memoria hasta ahora analizadas son conformadas por información de un tipo u otro, pero el contenido también puede ser interpretado como un código, solicitando una operación que se requiere de la computadora.

Considere el sencillo ejemplo de una suma binaria en donde solicitamos que la palabra almacenada en cierta dirección de memoria sea sumada a otra, contenida en una dirección distinta, y que el resultado sea almacenado en una tercera dirección. Los pasos necesarios para lograr esta suma serían:

1. Identificar la dirección de la primera palabra a sumar.
2. Transferir el contenido de la dirección a los registros provisionales de almacenamiento dentro de lo que forma la Unidad de Procesamiento Central.
3. Identificar la dirección de la segunda palabra a sumar.
4. Sumar el contenido de esta última palabra a la que se encontraba en los registros de la ***UPC***, obtenida en el paso 2, por medio de la Unidad Lógica y Aritmética (***UAL***).
5. Identificar la dirección de memoria donde el resultado se almacenará.
6. Transferir la suma almacenada en el registro adecuado de la ***UPC*** a la dirección obtenida en el quinto paso.

Ilustrar el concepto de un programa de computadora con estos seis sencillos pasos es sólo el comienzo, Tendremos que responder a lo largo del libro a las preguntas: ¿Cómo realiza la computadora las operaciones requeridas por los códigos de instrucciones que conforman el programa? ¿Qué es lo que la computadora demanda de la lógica externa para poder realizar su trabajo? ¿Cómo se escribe un programa de computadora?

11.4 Componentes de la Unidad de Proceso Central

Si consideramos el sencillo programa de suma descrito en la sección *11.1.4*, podemos examinar algunos de los requerimientos internos a la ***UPC*** para poder satisfacer lo que se le solicita por medio del código de instrucción que llamaremos ***SUMA***.

11.4.1 Registros

Como ya describimos al analizar la Unidad Aritmética y Lógica (***UAL***), es necesario un almacenamiento provisional dónde almacenar los números para trabajar con ellos, así como un registro dónde almacenar el resultado de las operaciones que a partir de ahora llamaremos ***acumulador*** y abreviaremos con *A*. En el acumulador podemos almacenar información tanto antes de operar con ella, como el resultado de las operaciones con los datos binarios. Por el momento y para simplificar las cosas consideremos que tenemos sólo un acumulador y que éste es de 8 bits[38].

Se requiere de este tipo de almacenamiento provisional por dos razones:

A. El contenido de la memoria sólo se puede leer o escribir; no es posible operar sobre los registros de la memoria principal.
B. Es posible tomar los datos directamente de memoria, operar sobre ellos y regresar el resultado otra vez a la memoria principal, pero una cantidad adicional importante innecesaria de circuitos lógicos serían requeridos para esto, complicando el diseño, subiendo el costo y

[38] Así como el número de transistores por mm^2 ha aumentado a lo largo de los años, el número de bits que maneja el acumulador también ha evolucionado pasando paulatinamente de 8 a 16 llegando hasta los 32 y 64 bits actuales. Así mismo se ha pasado de un sólo acumulador a varios de ellos abandonándose el concepto de acumulador único y pasando al de registros de propósito general (16 o más de ellos) cuya función es, en esencia, más o menos similar al del acumulador.

disminuyendo la rapidez de las operaciones como será aparente según avancemos en el capítulo.

Para poder acceder a una palabra de memoria, tanto para leer como para escribir su contenido, necesitamos tener su dirección y ésta se almacena en un registro especial que llamaremos ***registro contador de datos*** (data counter register) que abreviaremos con sus siglas en inglés: ***DC***.

El tamaño de este registro depende de la cantidad de memoria principal a la que la computadora pueda acceder directamente. En la figura *11.3* suponemos un registro contador de datos de 16 bits que nos da una capacidad de memoria de 65,536 palabras o 64K bytes (2^{16}; si la palabra es de 1 byte).

Una computadora puede constar de más de un registro de este tipo, pero una vez más, por simplicidad, sólo consideraremos uno.

Necesitamos también un registro donde la instrucción sea contenida de forma que la ***UPC*** pueda hacer referencia a ese registro para ver de qué instrucción, de las muchas de una computadora, se trata. A este tipo de registro se le denomina ***registro de instrucción*** (instruction register) que abreviaremos con la letra ***I***.

La dirección de la memoria de la que traeremos la siguiente instrucción (es nuestro ejemplo la de ***SUMA***) es almacenada en el llamado ***contador de programa*** (program counter) que abreviaremos con sus siglas en inglés: ***PC***.

El contador de programa es análogo al de datos, pero se asume que uno contiene la dirección de los datos en memoria y el otro la dirección de la siguiente instrucción en memoria. Para explicar un poco más este punto: si la operación ***SUMA*** consta de un código de operación que sólo ocupa una palabra y le deben seguir dos palabras más de memoria que contengan los datos a sumar, el apuntador a la siguiente instrucción debe actualizarse sumando tres localidades para que salte los datos y apunte efectivamente a la siguiente instrucción; de forma similar, el contador de datos se actualizará sumando 1 al registro de contador de programa y luego 1 más para poder localizar el siguiente dato.

Como se puede ver de la explicación anterior, existe una diferencia conceptual grande entre el contador de programa y el contador de datos, pues mientras el contador de programa debe

incrementarse continuamente para encontrar la siguiente instrucción usando para ello la longitud en palabras de que consta la instrucción anterior. El contador de datos puede o no incrementarse de acuerdo con si la instrucción requiere de datos o no.

Debe, por lo tanto, permitírsele gran flexibilidad al programador para poder modificar el registro contador de datos, mientras que, usualmente, al registro contador de programas no se tiene acceso más que de forma indirecta.

Figura 11.113 Registros básicos de la UPC.

	8 bits	8 bits
A	Acumulador	
DC	Contador de datos	
PC	Contador de programas	
I	Reg. de Instrucción	
	Otros	

11.4.2 Forma de Usar los Registros

Para poder comprender en su totalidad el uso de los distintos registros que forman parte de la ***UPC***, realizaremos paso a paso una suma como se describe en el algoritmo de la sección *11.1.4*.

Para realizar la suma asumiremos que la ***UPC*** puede interpretar 3 instrucciones sencillas: ***CARGA***, ***SUMA*** y ***GUARDA***. Cada una de estas instrucciones está definida de antemano rigurosamente y se deben seguir los pasos que el fabricante explica. Supondremos que las instrucciones predefinidas por el fabricante realizan estos pasos:

- SUMA: Toma el contenido de la dirección formada por el contenido de las siguientes dos palabras y la suma a la que está ya en el Acumulador, dejando el resultado en el Acumulador. Inventamos el código de instrucción 30 hexadecimal que, al ser encontrado por la ***UPC*** en una localidad de memoria, ésta la interpreta como la instrucción ***SUMA***.
- CARGA: Toma el contenido de la dirección formada por el contenido de las siguientes dos palabras y la guarda en el Acumulador. Inventamos el código de instrucción 40 hexadecimal que al ser encontrado por la ***UPC*** en una localidad de memoria éste la interpreta como la instrucción ***CARGA***.

- GUARDA: Toma el contenido del Acumulador y lo guarda en la dirección de memoria formada por el contenido de las siguientes dos palabras. Inventamos el código de instrucción 50 hexadecimal que al ser encontrado por la ***UPC*** en una localidad de memoria ésta la interpreta como la instrucción ***GUARDA***.

Recuerde que el Contador de Programa, ***PC***, debe incrementarse continuamente y de la forma adecuada para apuntar a la siguiente dirección de memoria donde se almacena una instrucción o dato.

Nuestro programa de suma sería entonces (las direcciones y datos están en hexadecimal):

```
CARGA  E0E0
SUMA   E0E1
GUARDA E0E2
```

Notemos que aquí se introduce un nuevo concepto llamado ***mnemónico*** que nos auxilia a recordar qué realiza cada código de instrucciones de la computadora en cuestión; pues en la palabra de memoria no se guardan las letras de la instrucción sino un simple código que representa a la instrucción:

```
40 E0 E0
30 E0 E1
50 E0 E2
```

Recordemos a su vez, que cada número en hexadecimal tiene su equivalente en binario y a su vez, cada número en binario se resume en si hay corriente o no en cada uno de los biestables que conforman el registro en cuestión, ya sea de memoria o interno a la ***UPC***.

Coloquemos estos códigos en un mapa de memoria en una localidad arbitraria, por ejemplo, a partir de la dirección C0 00_{16} hexadecimal (suponemos que tenemos una memoria total de 64K bytes — *FF* FF_{16} — por lo que basta con 16 líneas de dirección, 2^{16}, 16 bits binarios o 4 hexadecimales para conocer su localización). En la parte izquierda de la figura *11.4* esquematizamos la forma en que queda cargado nuestro pequeño programa en memoria, de forma aún desconocida, a partir de esa dirección.

Note que quedan definidas dos claras zonas en la memoria:

- Una del programa que va de *C0* 00_{16} a *C0* 08_{16}.

- Una de datos del programa que va de *E0* $E0_{16}$ a *E0* $E2_{16}$.

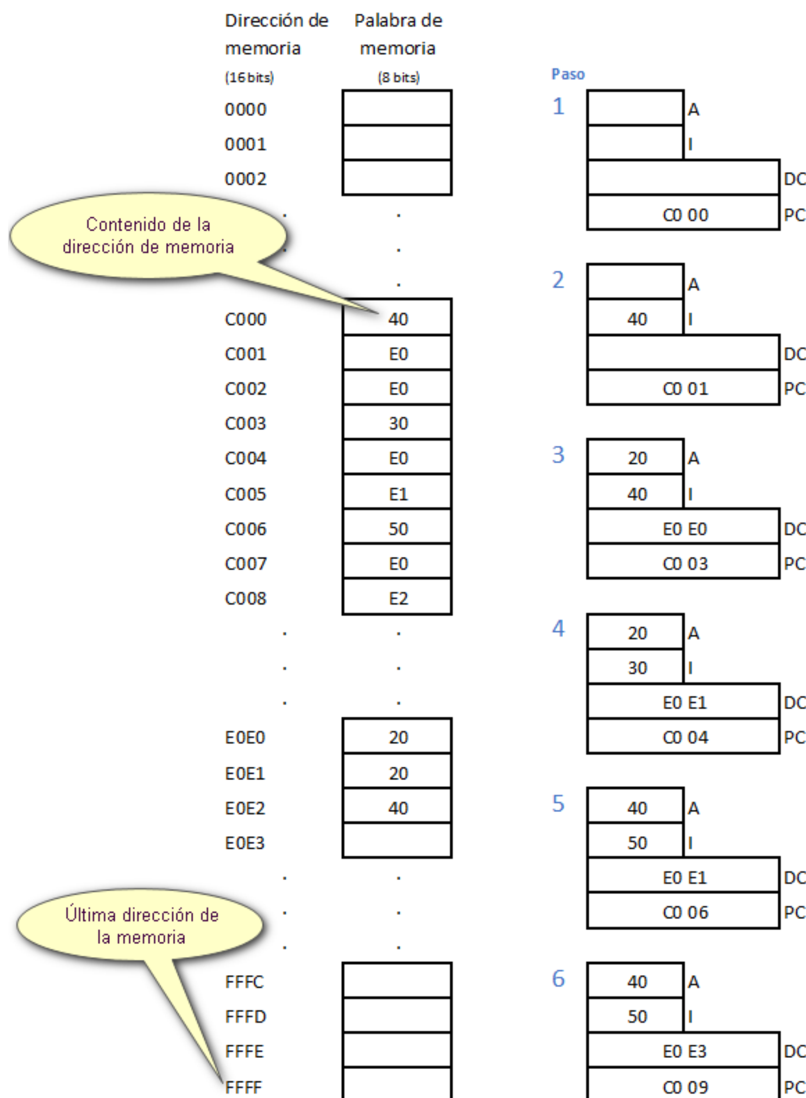


Figura 11.114 Uso de la memoria y registros.

Podemos seguir paso a paso el contenido de los registros hasta ahora analizados en la siguiente secuencia (se muestra el contenido de los registros en la parte derecha de la figura *11.4*):

1. De una forma, por ahora desconocida para nosotros, el registro del contador de programa o *PC* contiene la dirección de la primera instrucción: C0 00_{16}.
2. El ***UPC*** carga el contenido de la palabra de memoria apuntada por el ***PC*** al registro de instrucciones o ***I***, en este caso el número 40 hexadecimal (***CARGA***), e incrementa en uno el contenido del registro ***PC***.

3. El código 40 que aparece en el registro de instrucciones hace que la lógica de la ***UPC*** realice dos pasos. Primero, el contenido de la palabra de memoria indicada (direccionada) por el registro ***PC*** se trae de la memoria, pero se almacena en el bit más significativo del registro contador de datos (***DC***). La Unidad de Procesamiento Central incrementa posteriormente el contenido del ***PC*** en uno. El segundo paso consiste en traer el contenido de la palabra de memoria apuntado por ***PC*** y guardado en el bit menos significativo del registro ***DC***. Ahora, se utiliza el contenido del registro ***DC*** como apuntador para traer la palabra de memoria indicada al registro acumulador (***A***). Una vez más, la ***UPC*** incrementa el contenido del registro ***PC*** en uno. Con esto la instrucción número 1 (***CARGA***) se completa. Nótese que una instrucción consumió varios ciclos de reloj internos lo que nos da una primera indicación de la velocidad de la ***UPC***.
4. Una vez terminada la instrucción anterior la ***UPC*** trae el contenido de la palabra de memoria direccionado por el registro ***PC*** e incrementa en uno este registro. Puesto que no le hemos dado ninguna otra instrucción específica a la ***UPC***, el contenido de la palabra que se trae de la memoria se almacena en el registro ***I*** para ser interpretado como una instrucción.
5. Algo muy similar al paso 3 sucede en esta etapa. El registro ***DC*** contiene ahora la dirección de la palabra de memoria a sumar (E0 E1) y la ***UPC*** pasa el contenido de esta palabra y la del acumulador a la ***UAL*** para su suma; el resultado se almacena en el acumulador. Al final de la instrucción el contador de programa se ha incrementado en 2 para quedar apuntando a la siguiente instrucción.
6. La última instrucción de nuestro pequeño programa es algo muy similar a la instrucción 1 pero en lugar de traer los datos de la dirección apuntada por el registro ***DC***, se toma el contenido del Acumulador para almacenarlo en esa dirección.

11.4.3 Banderas de Estado

Muchas veces es necesario saber en detalle qué es lo que realiza la ***UAL*** para poder decidir si la operación que se ordenó se completó exitosamente o no, o para tomar ciertas medidas con el resultado que nos entrega la ***UAL***.

A los registros especializados en decirnos el resultado de las operaciones se les llama ***registros de estado*** (estatus register) o banderas y consisten en un flip–flop de un bit que nos indica varias situaciones dentro de la ***UAL***.

Algunas de estas situaciones en las que las banderas son modificadas, que nos son de interés, se resumen en los siguientes puntos:

1. El resultado fue cero.
2. Hay un acarreo.
3. Hay signo en la operación.
4. Hubo un desbordamiento en la operación.
5. Indicador de paridad.
6. Si la operación a realizar es suma o resta.
7. Acarreo a la mitad de los bits que forman la palabra.

Aquellas instrucciones que afectan o no a las banderas de estado son cuidadosamente seleccionadas por el fabricante y varían de computadora a computadora.

11.5 Ejecución de Instrucciones

Como en casi toda la lógica digital, la ***UPC*** es controlada por un reloj al que nos referiremos de ahora en adelante con el signo Φ. Este reloj puede ser tan sencillo como una sola señal o la combinación de varias señales de reloj requeridas por la ***UPC*** (refiérase a la figura *11.5a*).

La ejecución de una instrucción puede ser dividida en dos partes principales:

- Traer la instrucción (*Fetch* en inglés)
- Ejecutar la instrucción (*Execute* en inglés)

Si recordamos nuestro ejemplo anterior de un programa, cada instrucción comienza con el código que se carga en el registro de instrucción (***I***). A esta operación le llamamos traer la instrucción.

Durante la parte donde se trae la instrucción, la ***UPC*** tiene que colocar el contenido del registro del contador de programa (***PC***) en el bus de direcciones junto con las señales necesarias para informar a la lógica exterior que se trata de una lectura a la palabra de memoria indicada por el contenido del bus de direcciones. Por lo

que a la lógica externa concierne, esto se trata de una simple lectura.

Mientras la lógica externa responde, la ***UPC*** usa su lógica interna para sumar 1 al contenido del registro ***PC***. El contador de programas ahora apunta a la siguiente localidad de memoria de la que el código de la instrucción se leyó.

Una vez que el código de instrucción se encuentra en el registro de instrucciones, se dispara una serie de eventos controlados por la Unidad de Control que constituyen la ejecución de la instrucción.

En su forma más sencilla, dos periodos de reloj son necesarios para ejecutar la instrucción. Uno de ellos marca el tiempo para traer la instrucción y el otro para ejecutarla (figura *11.5c*).

Si consideramos las interconexiones necesarias para esto, necesitaremos hasta el momento, por lo menos las siguientes líneas de conexión:

1. Las líneas necesarias para la dirección de memoria. Por simplicidad consideraremos un diseño en el que se puedan acceder únicamente 65,535 localidades distintas de memoria o 64K bytes para lo que se requiere de 16 líneas (2^{16}). En conjunto forman el llamado bus de direcciones.
2. Las líneas necesarias para leer o escribir información a la memoria. Una vez más, por simplicidad, consideraremos un diseño de 8 bits de longitud de palabra por lo que se requieren de 8 líneas de datos que en conjunto forman el llamado bus de datos (2^{8}).
3. La alimentación del circuito/circuitos que forma la ***UPC*** formado como mínimo con una línea de voltaje y otra de tierra.
4. Las señales adecuadas para informar a la lógica externa que se trata de una lectura o escritura a la memoria.
5. La línea del reloj.

Mostramos en la figura *11.5c* el diagrama parcial de un circuito integrado de una ***UPC*** ficticia con 20 patillas externas.

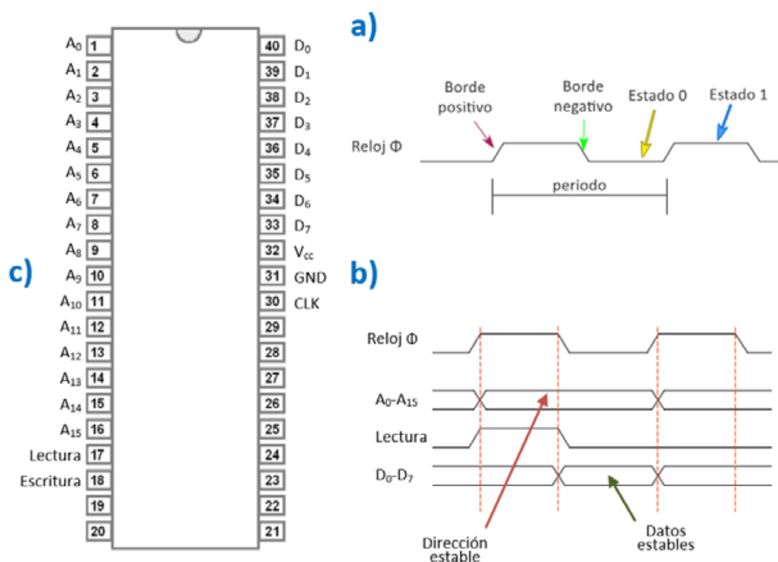


Figura 11.115 Interconexiones y diagrama de tiempos de una UPC.

El diagrama de tiempos que se presenta en la figura *11.4b* define la secuencia de traer una instrucción a la ***UPC***.

Algunas veces se requiere de más líneas que las conexiones que físicamente permite el empaque[39]. Se utiliza entonces una técnica llamada ***multiplexión*** en la que las mismas interconexiones externas sirven para varios propósitos. Por ejemplo, si se requieren de 20 líneas de dirección y sólo tenemos disponibles 14 conexiones, podemos mandar en el momento ***t*** la parte baja de la dirección por las 14 conexiones disponibles y las restantes 6 faltantes en el momento *t+1* por las mismas interconexiones disponibles.

11.6 La Unidad de Control

La IEEE define a la Unidad de Control como aquella parte de una computadora digital que se encarga de leer las instrucciones en secuencia, vigilar su aplicación y generar los comandos adecuados para que la ***UAL*** y los circuitos auxiliares la interpreten correctamente.

Examinaremos ahora cómo la Unidad de Control decodifica las instrucciones dentro de la Unidad de Procesamiento Central.

---

[39] A la fecha de edición de este libro ciertos microcomputadores requieren de 1200 interconexiones en su zoclo. Compare con las 16 patillas con las que contaba el que se considera como el 1er µprocesador comercial práctico: el Intel 4004 con palabra de 4 bits, instrucciones de 8 bits y dirección de 12 bits.

La unidad de control se puede representar funcionalmente como una caja negra tal como en la figura *11.2*, pero en realidad consiste en un gran número de elementos lógicos que se activan usando una secuencia de señales habilitadoras.

Latch

Elemento de memoria provisional usado para guardar información de forma temporal.

En la figura *11.6* mostramos un diagrama un poco más detallado de la Unidad de Procesamiento Central. El complementador, por ejemplo, es un elemento lógico que está disponible en todo tiempo para complementar el contenido de los latches. Una señal única de la Unidad de Control activa dicho elemento.

Sin embargo, el complementar el contenido de una memoria provisional no nos sirve de mucho. Lo que sí tiene valía es complementar el contenido del Acumulador. Esto implica mover la información al complementador y una vez realizada la operación, regresar el resultado al Acumulador.

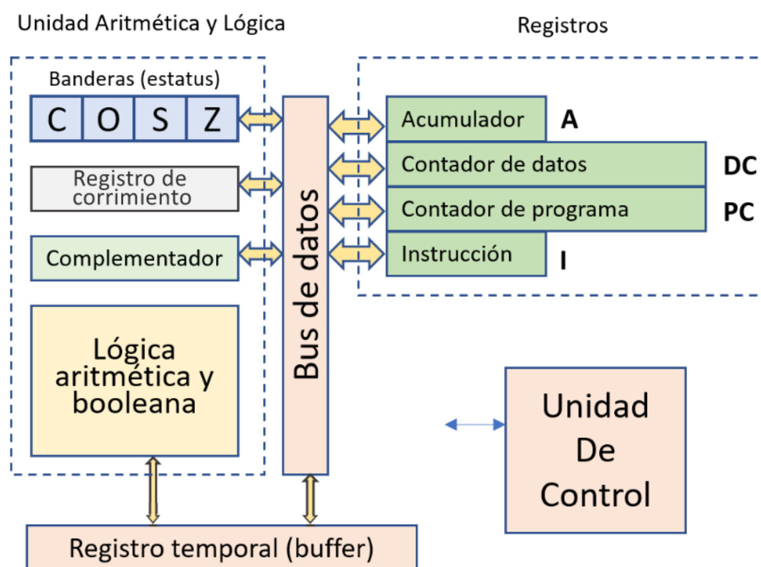


Figura 11.116 Detalle de la Unidad de Procesamiento Central.

Complementar el contenido del Acumulador requiere entonces 5 pasos:

1. Mover el contenido del Acumulador al Bus de Datos.
2. Mover el contenido del Bus de Datos al Complementador.
3. Activar la lógica del Complementador.
4. Mover el contenido del Complementador al Bus de Datos.
5. Mover el contenido del Bus de Datos al Acumulador.

A cada uno de estos 5 pasos se les llama ***microinstrucción***. Cada microinstrucción es habilitada por una señal de la Unidad de Control. Si la Unidad de Control manda la combinación de señales adecuadas, cualquier número de microinstrucciones pueden ser ejecutadas para responder a una ***macroinstrucción*** que es la respuesta aceptada de la Unidad de Procesamiento Central a un código de una instrucción de lenguaje de máquina; muy similares a las que describimos en la sección *11.2.1* y *11.2.2* (***SUMA***, ***CARGA*** y ***GUARDA***).

Para complementar el contenido del Acumulador en nuestro ejemplo previo, la Unidad de Control requirió 5 códigos binarios que dispararon las señales de control apropiadas. La secuencia de códigos binarios que residen en la Unidad de Control se llaman ***microprogramas*** y generar esta secuencia de códigos binarios se le conoce como ***microprogramación***.

Existe un paralelo muy cercano entre lo que es microprogramar y la programación en lenguaje de ensamblador. Un microprograma se guarda como una serie de códigos binarios dentro de la Unidad de Control. Un programa en ensamblador es guardado como una serie de códigos binarios dentro de la memoria principal (***RAM*** o ***ROM***). Al programa en ensamblador se le conoce con el nombre de ***macroprograma*** y a cada macroinstrucción le corresponden uno o más microprogramas dentro de la Unidad de Procesamiento Central que deben ser ejecutados para generar la respuesta esperada.

Un microprograma guardado en la Unidad de Control tiene una memoria para datos, que consiste en los registros de la ***UPC;*** además de almacenamiento interno a la misma Unidad de Control. Un macro programa también tiene un área de almacenamiento, ya sea provisional (***RAM***) o definitiva (***ROM***).

La complejidad de las operaciones asociadas con una macroinstrucción es función directa del tamaño del microprograma que la macroinstrucción inicia. Pero existe el límite impuesto por el tamaño físico que puede tener la Unidad de Control y esto se hace más patente en las microcomputadoras, donde toda la ***UPC*** se encuentra contenida en un sólo circuito.

La Unidad de Control de cada computadora no es más que un microprograma. Si se permite al usuario modificarlo, se dice que la computadora es microprogramable. Si la Unidad de Control se

diseña como parte del diseño lógico de la computadora y forma parte inalterable de la Unidad de Procesamiento Central, entonces, la computadora no es microprogramable.

11.6.1 Ejemplo de Unidad de Control

Como ejemplo diseñemos una Unidad de Control sencilla y arbitraria que genera las señales de control indicadas en la figura *11.7*.

Figura 11.117 Detalle de la Unidad de Procesamiento Central.

La Unidad de Control debe generar estas señales en respuesta a las macroinstrucciones. En las tablas *11.1*, *11.2* y *11.3* describimos con cierto detalle estas señales de control.

Tabla 11.1	
Señales de la Unidad de Control	
Señal	Función
C_0, C_1	C_0=0, C_1=0 No se mueven los datos del bus de datos o del registro de direcciones
	C_0=0, C_1=1 Se mueven los datos del bus de datos o del registro de direcciones
	C_0=1, C_1=0 Se mueven los datos al bus de datos o al registro de direcciones
	C_0=1, C_1=1 La instrucción queda atrapada dentro de la Unidad de Control (Vea ejemplo en tabla *11.4*)
C_2, C_3, C_4, C_5	Cuando C_0, C_1=0 o C_0=0, C_1=1 estas cuatro señales se decodifican para especificar el flujo de información como se especifica en la tabla *11.2*
C_6, C_7, C_8	Estas tres señales se decodifican para controlar las operaciones de la Unidad Aritmética y Lógica como se especifican en la tabla *11.3*
Lectura, escritura	Conexiones directas a la salida de la Unidad de Procesamiento Central
C, O, S, Z	Cuatro bits de estatus conectados a 4 bits de los datos de la Unidad de Control: Acarreo, Saturación. Signo y Cero
Φ	Señal de reloj conectada a la Unidad de Control
D_0–D_7	Bus de datos conectado a la Unidad de Procesamiento Central por donde salen y entran los datos
A_0–A_7	Líneas de dirección que salen de la Unidad de Procesamiento Central

Tabla 11.2				
Selección de flujo cuando C_0= C_1=1				
C_5	C_4	C_3	C_2	Función
0	0	0	0	Acumulador ↔ bus de datos
1	0	0	0	Contador de Datos byte alto ↔ bus de datos
0	1	0	0	Contador de Datos byte bajo ↔ bus de datos

1	1	0	0	Contador de Programa byte alto ↔ bus de datos
0	0	1	0	Contador de Programa byte bajo ↔ bus de datos
1	0	1	0	Registro de Instrucción ↔ bus de datos
0	1	1	0	Registro de Estatus ↔ bus de datos
1	1	1	0	Registro de Corrimiento ↔ bus de datos
0	0	0	1	Complementador ↔ bus de datos
1	0	0	1	Latch ALU ↔ bus de datos
0	1	0	1	Buffer ALU ↔ bus de datos
1	1	0	1	Registro de Datos ↔ bus de datos
0	0	1	1	Contador de Datos al registro de dirección
1	0	1	1	Contador de Programa al registro de dirección
0	1	1	1	Registro de Datos ↔ buffer de dirección
1	1	1	1	No se usa
Notas: Latch=Memoria intermedia de almacenamiento Búfer (Buffer)= Memoria rápida de almacenamiento provisional				

TABLA 11.3			
Señales de selección de la Unidad Aritmética y Lógica			
C_8	C_7	C_6	Función
0	0	0	Selecciona registro de corrimiento lógico
1	0	0	Selecciona lógica de complemento
0	1	0	Selecciona lógica de suma*
1	1	0	Selecciona lógica ***Y******
0	0	1	Selecciona lógica ***O******
1	0	1	Selecciona Lógica ***O EXCLUSIVA******
0	1	1	Incrementa el latch de la Unidad Aritmética y Lógica
1	1	1	No se usa
*La operación se realiza en el contenido del latch de la ***UAL*** y el registro Buffer apareciendo el resultado en los latches de la ***UAL***			

Las señales de control descritas en las tablas anteriores no permiten a la Unidad de Control generar todas las operaciones necesarias para soportar una instrucción en lenguaje ensamblador. Por ejemplo, no se ha hablado de cómo las señales de ***LECTURA*** y ***ESCRITURA*** son generadas, o de cómo los cuatro latches de estado ***C*** (acarreo), ***O*** (saturación), ***S*** (signo) y ***Z*** (cero) son manejados.

Crearemos como ejemplo un simple microprograma cuya función sea traer una instrucción de memoria y colocarla en el registro de Instrucción (***I***). Recordemos que cada ejecución de una instrucción comienza con traerla de una localidad de memoria apuntada por el Contador de Programa y colocarla en el registro de Instrucción para ser interpretada.

Tabla 11.4										
Microprograma para traer una instrucción de memoria										
#	C_8	C_7	C_6	C_5	C_4	C_3	C_2	C_1	C_0	Función
1	1	1	1	1	0	1	1	0	1	Mover Contador de Programa a Registro de Dirección
2	1	0	0	0	0	1	1	1	1	Poner flip-flop de Lectura en 1, flip-flop Escritura en 0
3	1	1	1	0	0	1	0	0	1	Mover byte bajo del Contador de Programa al bus de datos
4	1	1	1	1	0	0	1	1	0	Mover el bus de datos al latch de la Unidad Aritmética y Lógica
5	0	1	1	0	0	0	0	0	0	Incrementar el latch de la Unidad Aritmética y Lógica
6	1	1	1	1	0	0	1	0	1	Mover latch de la Unidad Aritmética y Lógica al bus de datos
7	1	1	1	0	0	1	0	1	0	Mover bus de datos a Contador de Programa byte bajo
8	1	1	1	1	1	0	0	0	1	Mover Contador de Programa byte parte alta a bus de datos
9	1	1	1	1	0	0	1	1	0	Mover bus de datos al latch de la Unidad Aritmética y Lógica
10	1	0	0	0	0	1	0	1	1	Si estatus de acarreo es 0, salta siguiente instrucción
11	0	1	1	0	0	0	0	0	0	Incrementar el latch de la Unidad Aritmética y Lógica
12	1	1	1	1	0	0	1	0	1	Mover el latch de la Unidad Aritmética y Lógica al bus de datos
13	1	1	1	1	1	0	0	1	0	Mover bus de datos a Contador de Programa byte parte alta
14	1	1	1	1	1	0	1	0	1	Mover Registro de Datos a bus de datos
15	1	1	1	1	0	1	0	1	0	Mover bus de datos a Registros de Instrucción

Véase que cada microinstrucción se convierte en nueve bits binarios dentro de la Unidad de Control. Estos nueve bits no tienen nada que ver con la longitud de palabra de la computadora que en nuestro caso se escogió de 8 bits. La longitud de la palabra de la Unidad de Control se fija arbitrariamente a la longitud que sea necesaria para realizar sus funciones, en este caso nueve bits.

Nótese también que se requieren de 15 instrucciones para traer el código de instrucción de la memoria principal y esto debe ocurrir durante un pulso del reloj que sincroniza a la ***UPC.*** La Unidad de Control deberá dividir este pulso de reloj en 15 (para este caso) internos para ejecutar el microprograma paso a paso.

En la figura *11.8* mostramos la primera microinstrucción del microprograma de traer una instrucción de memoria y su significado.

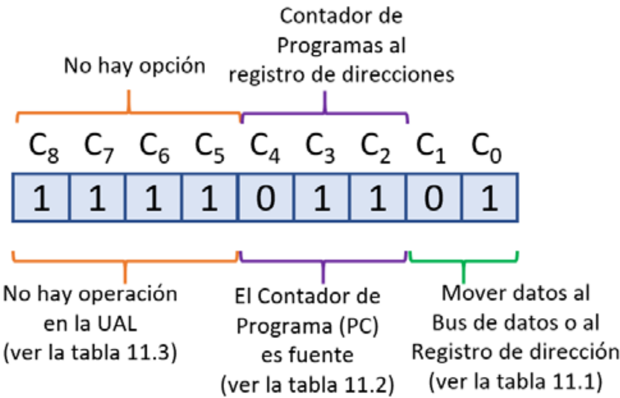


Figura 11.118 Construcción de una microinstrucción.

El mismo caso sucede con las demás microinstrucciones.

A continuación, hacemos una breve descripción del microprograma:

La primera microinstrucción mueve el contenido del Contador de Programas al Registro de Dirección que hace que el contenido del Contador de Programas aparezca en las líneas externas de dirección de la ***UPC.*** La microinstrucción 2 coloca la señal de ***LECTURA*** en verdadero y la de ***ESCRITURA*** en falso, indicándole a la lógica externa que el valor de las líneas de dirección (A_0 a A_{15}) es válido y que debe colocar el contenido de la localidad de memoria en las líneas de datos (D_0 a D_7). para ello tiene todo el tiempo que dure la interpretación de las microinstrucciones 3 hasta la 13, pues aún no se requiere los datos. Desde la microinstrucción 3 a la 13 se incrementa el Contador de Programas, *PC*, pero es necesario

realizarlo en dos partes pues los registros son de 8 bits mientras que el Contador de Programas es un registro de 16 bits. De la microinstrucción 3 a la 7 se incrementa la parte baja y de la 8 a la 13 la parte alta.

11.7 Resumen

La Unidad de Procesamiento Central (***UPC***) Forma el cerebro de la computadora y su parte central y principal que interpreta las instrucciones almacenadas en la memoria principal que conforman un programa. Sin la ***UPC*** no es posible el concepto de computadora tal como lo conocemos hoy en día.

La ***UPC*** se forma de la Unidad Aritmética y Lógica (***UAL***), los registros internos y la Unidad de Control (***UC***), existen además una serie de buses internos para llevar y traer información de los distintos componentes.

En este capítulo se introducen los registros y la Unidad de Control analizando en detalle cada componente y terminando con un ejemplo sencillo de una ***UC***.

11.7.1 Puntos Importantes del Capítulo

- La Unidad de Procesamiento Central (***UPC*** o ***CPU***) se forma de la ***UAL***, los registros y la Unidad de Control (***UC***).
- El contenido de la memoria principal puede ser interpretado como datos o instrucciones.
- Para saber si la información almacenada en la memoria es fidedigna, se utiliza la protección por distintas técnicas; una de las más usadas es la paridad que puede dividirse en par o impar.
- Los registros dependen del diseño de la máquina, pero en general se debe contar como mínimo con los siguientes: Acumulador, Registro de Instrucciones, Contador de Programa y Contador de Datos.
- El registro de Contador de programa es incrementado cada vez que una instrucción es traída a la ***UPC*** para su interpretación por la ***UC***.
- Existe también un registro dentro de la ***UPC*** donde se lleva el estado de varios indicadores importantes para un programador y para la misma ***UPC***.
- El ciclo de ejecución de una instrucción se divide en traer la instrucción (fetch) e interpretarla.

- En la Unidad de Control reside un programa llamado microprograma, que se encarga de interpretar las instrucciones almacenadas en la memoria principal llamadas macroinstrucciones.
- Existen ***UC*** microprogramables donde el programa que interpreta las macroinstrucciones puede cambiarse.
- Se debe dividir el reloj en ***n*** partes dentro de la ***UPC*** para poder ejecutar una microinstrucción en cada ciclo de reloj dividido.

11.8 Problemas

11.1 Calcule la probabilidad de error de la detección de errores por el método de paridad en una palabra de 8 bits donde se agrega un bit más de paridad.

11.2 Investigue la corrección de errores por medio del código de Hamming.

11.3 Investigue cuál es el código de la instrucción ***SUMA*** en por lo menos dos tipos distintos de computadoras o microcomputadoras.

11.4 De un circuito de microcomputadora cualquiera averigüe (no use uno muy reciente) toda su arquitectura interna y realice un croquis marcando claramente cada área de la ***UPC***, registros, ***UAL***, acumuladores, buses, unidad de control, etc.

11.5 Realice un microprograma para la Unidad de Control descrita en la sección *11.4.1* para la macroinstrucción ***SUMA.***

12 Lógica más allá de la UPC

En este capítulo identificaremos la lógica adicional que se requiere, además de la Unidad de Procesamiento Central y la memoria para generar un sistema lo suficientemente amplio para ser de utilidad.

Debemos identificar por separado cada uno de los componentes de un sistema de computación por su función, tal como ya lo hemos hecho con la memoria ***RAM*** y ***ROM***; pero no existe una correlación fundamentalmente necesaria entre el componente individual lógico y el circuito que realiza la función.

12.1 Entrada/Salida

La transferencia de datos entre la lógica que forma parte del sistema de computadora y aquella que está más allá de éste, se conoce en general como ***entrada/salida*** o ***E/S*** (input/output o I/O).

Se incluye dentro del ámbito del sistema a toda la lógica que se haya diseñado para trabajar en conjunción con la ***UPC***. Toda aquella lógica que no caiga dentro de esta clasificación se le conoce como externa.

La interfaz entre el sistema de computación y la lógica externa debe estar claramente definida; debe proveer facilidades para transferir datos, además de las señales de control que identifiquen a todos los eventos que ocurren.

Hay muchas formas en las que un sistema puede transferir datos hacia el exterior; pero todos caen en las siguientes tres categorías:

1) ***E/S*** Programada. En este caso todas las transferencias de datos entre la computadora y el exterior son controladas por la computadora o, para ser más precisos, por un programa corriendo en la computadora. Existirá un protocolo bien definido en el que la computadora haga conocer a la lógica externa que los datos están disponibles en una localidad de memoria fija de donde ésta los pueda tomar, o bien, la computadora indicará de alguna forma a la lógica externa que está esperando que ponga información en ciertas localidades de memoria para poder accederlas. La característica clave de la Entrada/Salida Programada es

que la lógica externa hace exactamente lo que se le dice que haga.

2) ***E/S*** por Interrupción. Las interrupciones son una forma que tiene la lógica externa para forzar a la computadora a poner atención y suspender lo que está haciendo para atender a las necesidades de la lógica externa.
3) Acceso Directo a Memoria. Ésta es una forma de transferir datos entre la memoria interna y los dispositivos externos sin involucrar a la Unidad de Procesamiento Central en la lógica de la transferencia.

12.2 Entrada/Salida Programada

Los datos son transferidos entre el sistema de computación y la lógica externa, en cualquier dirección, por medio de un puerto de entrada y salida. Un puerto de entrada y salida consiste en un puerto con memoria de entrada y salida (buffer) conectado a las líneas de datos del bus del sistema y a las interconexiones que acceden a la lógica externa (ver figura *12.1*).

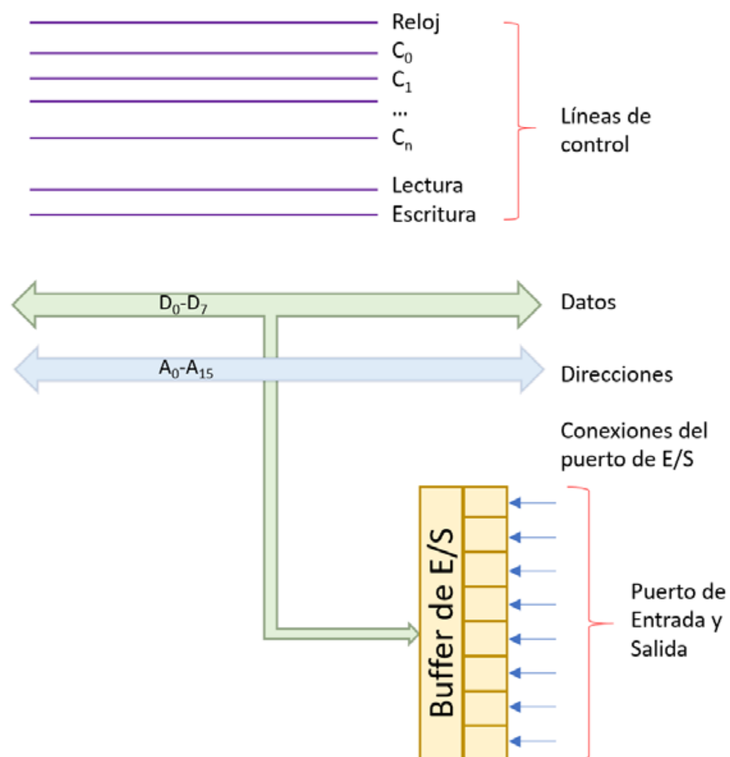


Figura 12.119 Puerto de entrada y salida.

Cuando la lógica externa transmite datos al sistema de cómputo, lo hace presentando los datos en las conexiones del puerto de

entrada y salida, por lo que este mismo puerto almacena provisionalmente estos valores. El puerto de entrada y salida no puede estar comunicándose constantemente con las líneas del bus de datos, puesto que estas líneas pueden estar llevando información de o hacia la memoria. Si las conexiones del puerto de entrada y salida se comunicaran permanentemente con el bus de datos, cada vez que la lógica externa presentase datos a las conexiones de entrada y salida, los datos se propagarían por las líneas del bus de datos con consecuencias impredecibles. La ***UPC*** debe, por lo tanto, seleccionar un puerto de ***E/S*** y leer el contenido del puerto en la misma forma que lo hace de una dirección de memoria.

Como los datos se leen de los puertos de ***E/S*** tal como si se tratase de una localidad de memoria, a esta forma de transferencia de ***E/S*** se le conoce también como ***E/S de mapa de memoria*** (memory mapped I/O). Este tipo de esquema permite a la computadora utilizar las mismas instrucciones poderosas de lectura y escritura tanto para localidades de memoria como para los puertos de Entrada/Salida. Por ejemplo, se pueden realizar operaciones aritméticas directamente con los puertos sin necesidad de almacenar los datos en memorias provisionales.

Normalmente existe más de un dispositivo externo por lo que debe haber más de un puerto de entrada y salida. Una forma rudimentaria, pero muy utilizada en sistemas pequeños, de obtener uno o más puertos de entrada y salida consiste en dividir las líneas de dirección de la ***UPC*** entre la memoria y los puertos. Por ejemplo, si tomamos la línea A_{15} de un sistema de microcomputadora de 64Kb y escogemos que cuando su valor sea 0 se acceda a la memoria y que cuando éste sea 1 se acceda a los puertos de entrada y salida, tendremos dividida la memoria en dos áreas bien diferenciadas; una de las cuales nos sirve para comunicarnos con los dispositivos externos. En la figura *12.2a* mostramos el esquema de un dispositivo de ***E/S*** paralelo con un puerto.



Figura 12.120 Dispositivos de interfaz de E/S paralelos.

A los dispositivos de la figura *12.2* se les conoce como dispositivos de entrada y salida en paralelo puesto que los datos son escritos y leídos en una palabra completa a la vez. Nótese que no hay ninguna razón por la que el dispositivo de entrada y salida sólo deba tener un puerto de entrada y salida. En la figura *12.2b* mostramos un dispositivo de ***E/S*** paralelo de dos puertos. El número de puertos de entrada y salida que un dispositivo paralelo de ***E/S*** tiene, solamente es función de cuántas interconexiones estén disponibles económicamente en el empaque en que se coloque el dispositivo. El inconveniente de utilizar una línea de dirección para

seleccionar el buffer de un puerto de entrada y salida, es que el mapa de memoria se reduce automáticamente a la mitad; lo que, en la mayoría de los sistemas, de medios a grandes, es un precio muy severo a pagar, solamente para seleccionar un puerto.

Un método preferido es agregar a la ***UPC*** dos líneas más que controlan la transferencia de y hacia los puertos de salida. Una de ellas llamada ***IOSEL*** especifica que la dirección, que a continuación aparece en el bus de direcciones, es válida para una transferencia a los puertos de entrada y salida; la otra, llamada ***IORW***, indica si se trata de una lectura o escritura de los puertos de ***E/S*** (ver figura *12.3*).

Desafortunadamente la transferencia a ciegas de datos de un sistema de computación y la lógica externa, no siempre proveen la suficiente capacidad de entrada y salida. Las siguientes características no existen en este sistema:

1. El sistema de cómputo debe decir a la lógica externa cuando los datos han sido colocados en el buffer de entrada y salida y, por lo tanto, cuando están listos para ser leídos; de la misma forma la lógica externa debe indicar a la computadora cuando ha colocado datos en el buffer de entrada y salida, para que estos puedan ser leídos.
2. Tanto la computadora como la lógica externa deben tener alguna forma de informarse entre ellas de la naturaleza de los datos que van a intercambiar. Claramente los datos que se están transfiriendo pueden ser sujetos a varias interpretaciones; por ejemplo, pueden ser datos puros binarios, algún código que identifique alguna operación a ejecutarse, ser parte de una dirección o ser una dirección completa.

Cuando la computadora transmite señales a la lógica externa como un medio para identificar eventos o datos, se refiere a estas señales como controles de entrada y salida. La misma información viajando en sentido opuesto, esto es de la lógica externa hacia la computadora, se le conoce como estado de entrada y salida. La lógica externa no puede controlar a la computadora; sólo puede enviar información de su estado para que la computadora la interprete.

Los sistemas de computación, usualmente, tienen toda una serie de controles de entrada y salida, así como líneas de estado que

están separadas y diferenciadas de los puertos de entrada y salida. Por lo general las computadoras asignan uno o más puertos de entrada y salida para funcionar como conductores del control y del estado, mientras que otros puertos de entrada y salida son los encargados de transferir los datos. Es solamente la forma en que la ***UPC*** interpreta los datos que hace la diferencia entre información de datos, control o estado de los dispositivos.

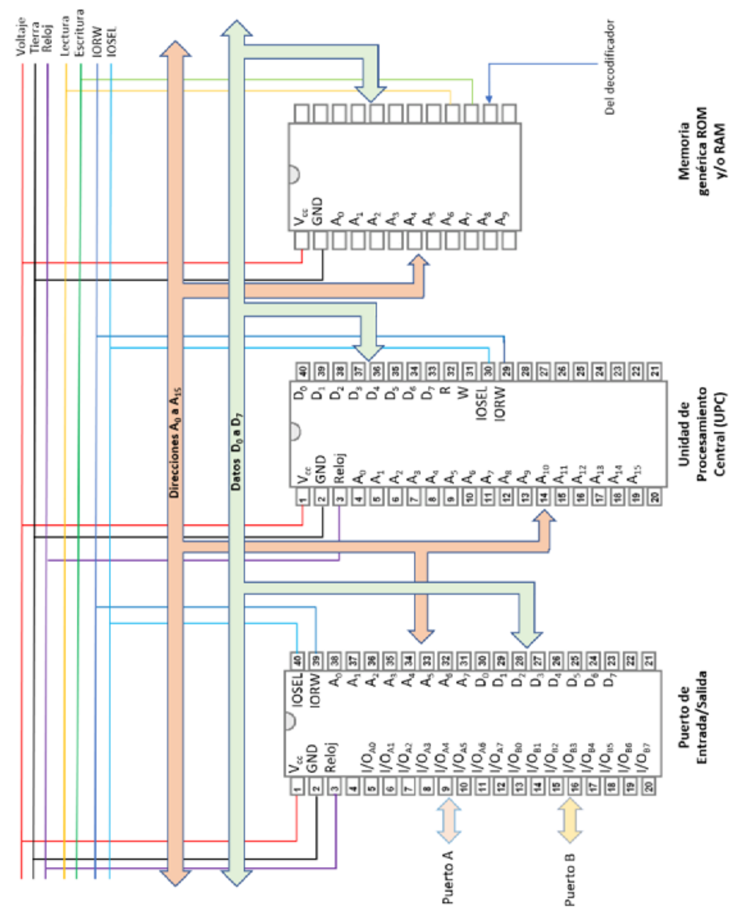


Figura 12.121 Circuito paralelo de E/S con lógica de dirección.

12.3 Entrada/Salida por Interrupción

Casi todas las ***UPC*** tienen una línea por la cual la lógica externa puede solicitar atención inmediata. A ésta se le conoce como ***señal de petición de interrupción*** puesto que por medio de ella la lógica externa pide que sea interrumpido lo que se esté haciendo y se le preste atención.

Comenzaremos por describir un ejemplo, que, aunque poco realista, contiene los elementos claves de los servicios a interrupciones.

Suponga que tenemos un sistema que atiende a la temperatura del agua de una regadera. Un sensor de temperatura informa constantemente a la ***UPC*** de los cambios en la temperatura del agua que sale por la regadera. La ***UPC***, por medio de un programa, compara la temperatura actual con una fija y manda señales adecuadas que deben ser interpretadas como comandos para que cierren o abran las válvulas de agua fría o caliente según se necesite. Debemos tener un programa no trivial que llamaremos ***AJUSTA*** que realice los cambios necesarios de una forma racional y que no intente hacer ajustes descabellados. El programa quedaría localizado en el mapa de memoria como se muestra en la figura *12.4a*.

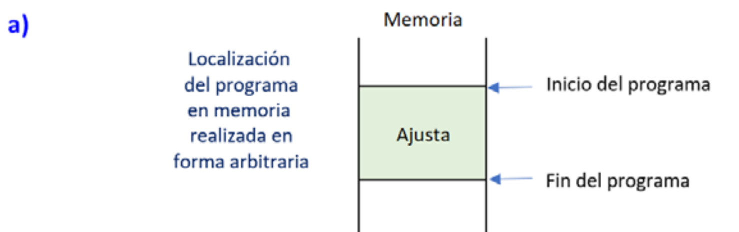
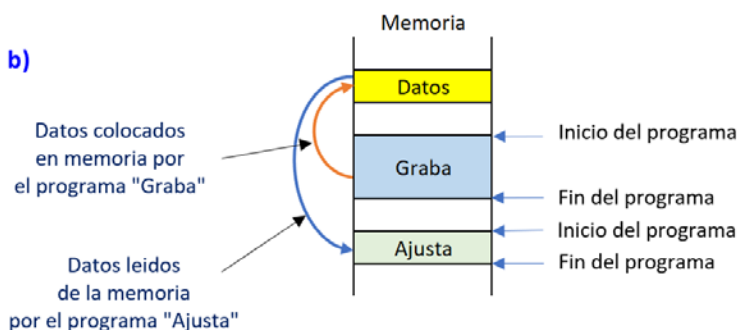


Figura 12.122 Programas que comparten la misma área de datos.



Otro programa llamado ***GRABA*** leería los datos del sensor y los escribiría en un área de memoria conocida por el programa ***AJUSTA***. El único contacto entre los dos programas sería el hecho de que uno coloca los datos en un área de memoria (***GRABA***) y el otro los lee de la misma área de memoria (ver figura *12.4b*).

La forma en que el sensor de temperatura lee los datos y los envía a la computadora es otra parte del problema no trivial. Un sensor de temperatura barato puede realizar lecturas aproximadamente

cada quinto de segundo; lecturas que deben ser convertidas de su forma analógica a digital (explicado en el capítulo *16*). Una computadora lenta pude realizar alrededor de 500,000 operaciones por segundo por lo que es muy probable que, si intentamos mandar las señales obtenidas del sensor de temperatura a un puerto de ***E/S***, la computadora no las leerá en el momento adecuado y se perderán hasta la siguiente lectura.

Una secuencia ordenada de eventos para atraer la atención de la ***UPC*** nos permite resolver esta contingencia:

1) La lógica del sensor de temperatura transmite una señal de ***petición de interrupción*** (Interrupt Request Signal, ***IREQ***) vía una línea de control del bus.
2) La ***UPC*** tiene la opción de aceptar o rechazar esta señal de interrupción; la acepta mandando una señal a su vez por el bus de control que indica que la interrupción ha sido aceptada y ésta es llamada ***señal de aceptación de interrupción*** (interrupt acknowledge signal, ***IACK***). La señal se transmite por medio del bus de control de la computadora.
3) El dispositivo externo usa la señal de aceptación de interrupción como señal habilitadora, transmite sus datos a un puerto de ***E/S*** y remueve la señal de petición de interrupción indicando con esto que ya no requiere atención.

El diagrama de tiempo de estas acciones se muestra en la figura *12.5a*.

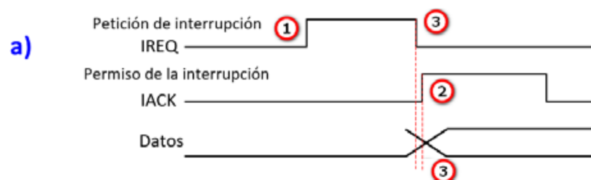
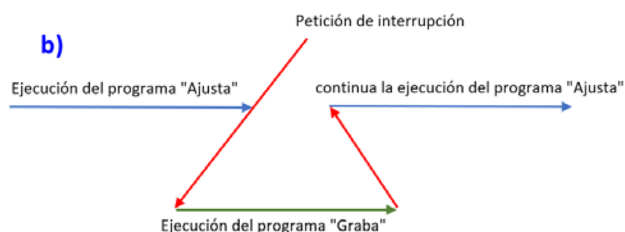


Figura 12.123 Secuencia de una interrupción.



Nótese que, aunque nos hemos referido a la lógica externa entregando datos, el flujo de información bien puede ser en el sentido contrario. De hecho, nada nos dice que una salida o entrada de datos tenga que suceder después de una interrupción; bien pueden ser señales de control las que se transmitan.

El propósito de la interrupción es informar a la ***UPC*** que suspenda lo que está haciendo, procese los datos que se están transmitiendo desde el puerto de ***E/S*** y reanude lo que suspendió. En referencia a nuestro ejemplo con los programas ***GRABA*** y ***AJUSTA***, ocurre una serie de eventos similar a lo descrito en la figura *12.5b*.

La característica más importante de toda esta serie de eventos es que son asíncronos e impredecibles.

El diagrama de un sistema de interrupciones con las señales descritas se muestra en la figura *12.6*.

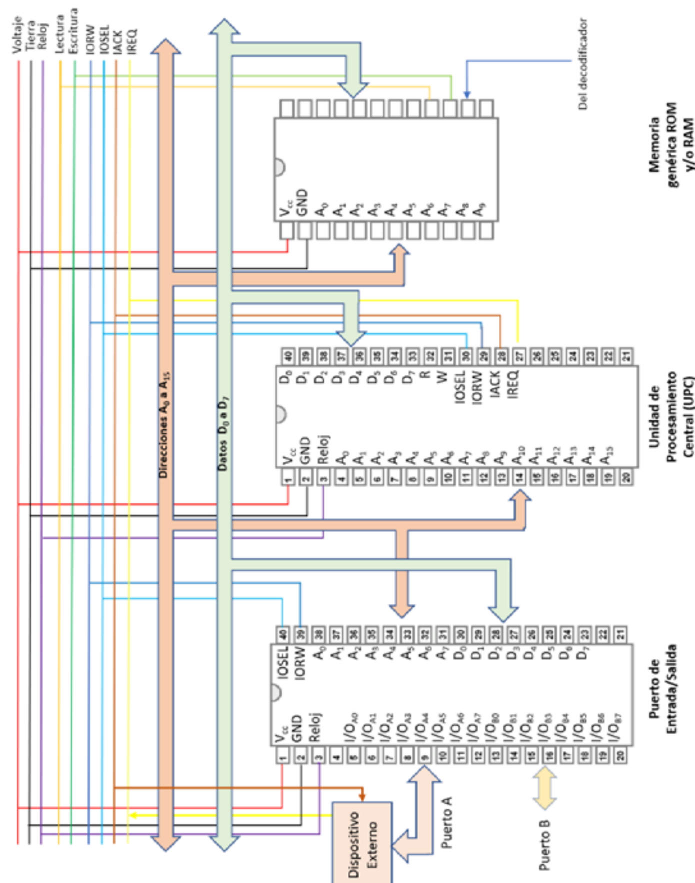


Figura 12.124 Dispositivo externo que usa interrupciones.

12.3.1 Respuesta de la UPC a una Interrupción

En su forma más elemental, la Unidad de Proceso Central puede responder a una interrupción cargando en el registro Contador de Programa (***PC***) la dirección inicial del programa que atiende a la interrupción. Esto nos deja con dos preguntas importantes:

- ¿Qué pasa con el programa que está en ejecución?
- ¿De dónde obtiene la ***UPC*** la dirección del programa que la lógica externa necesita que se ejecute?

El programa que se está ejecutando en memoria tiene información almacenada en los registros de la ***UPC*** que puede ser importante: el acumulador (***A***), el Contador de Programa (***PC***), el Contador de Datos (***DC***) y otros registros existentes en la ***UPC***. Si el nuevo programa a ejecutar no tiene algún mecanismo que guarde toda esta información, se corre el riesgo de no poder volver al punto donde estaba en ejecución el programa que fue interrumpido en primer lugar. Si guardamos los valores de los registros con sólo restituirlos a los registros correspondientes, en especial el Contador de Programa, podemos regresar al sitio en el que suspendimos la ejecución.

Una interrupción no se reconoce hasta terminar con la instrucción que se está ejecutando; por lo que el valor del registro de Instrucción (***I***) es quizá el único que no importe guardar, pues será inmediatamente substituido por la instrucción próxima a ejecutarse.

Existen dos formas de guardar la información necesaria para poder volver al punto donde se interrumpe un programa antes de comenzar el que atiende a la interrupción:

1) La ejecución de un microprograma, almacenado en la Unidad de Control, que guarde automáticamente los registros por nosotros en un área predeterminada de memoria. A esta área de memoria reservada se le llama ***Pila*** (Stack) (para su administración se reservan instrucciones de ensamblador, que se verán en más detalle en siguiente capítulo, especiales: ***POP*** (extraer) y ***PUSH*** (empujar); en algunos diseños de ***UPC*** se guardan en un juego de registros paralelos).
2) Dejar que el programador ejecute primero una ***rutina de servicio a la interrupción*** que se encargue de almacenar los datos de los registros en un lugar seguro.

Al final de la ejecución del programa de interrupción, no importando qué método esté disponible, se deben realizar los pasos en orden contrario para restaurar los registros correspondientes.

A continuación, consideremos la forma de obtener la dirección donde se ejecutará la rutina que atiende a la interrupción. Si se trata de un sólo caso como en nuestro ejemplo de la regadera, podemos considerar que al tener una dirección de memoria fija ya la conozca de antemano la ***UPC***, pero en el caso de múltiples puertos de entrada y salida esto resulta un poco impráctico.

Considere la forma en que la ***UPC*** responde a una interrupción:

1) Manda la señal de aceptación de la interrupción, ***IACK***.
2) Guarda de forma automática todos los registros o permitir al usuario guardarlos de alguna manera.
3) Mueve el contenido del ***vector de interrupciones***, que el puerto de ***E/S*** manda, al Contador de Programa para comenzar a ejecutar un programa en otra localidad de memoria.

Utilizar un vector de interrupciones que forme la dirección en memoria donde se encuentra el programa que atiende a la interrupción, proporcionado por el dispositivo que interrumpe, nos permite tener gran flexibilidad al momento de definir nuestras rutinas.

12.3.2 Código de Selección de un Dispositivo que Interrumpe

Otro de los esquemas muy utilizados consiste en exigir que el dispositivo que interrumpe a la ***UPC*** mande un código que identifique qué puerto de ***E/S*** es el que interrumpe.

Si se manda un código de selección del dispositivo, podemos entonces acceder a una tabla en memoria que nos dé el vector de dirección de interrupción que cargaremos al Contador de Programa (*PC*) para ejecutar la rutina de interrupción.

Para que esto funcione se exige un poco más de la lógica externa pues ésta debe ser capaz de almacenar su código de selección; para ello existen en el mercado varios tipos de circuitos programables de interfaz de ***E/S*** (***PIO***, ***PIA***, etc.)

Usando como referencia la figura *12.6* (la diferencia es que ahora el dispositivo externo se une a las líneas de datos del sistema para

mandar por ellas su código de identificación), la secuencia de eventos es como sigue:

1. La lógica del dispositivo externo crea una señal de petición de interrupción que transmite a la ***UPC*** como ***IREQ***.
2. Cuando la ***UPC*** está lista para dar servicio a la interrupción, responde con una señal de aceptación o ***IACK***.
3. Al recibir la señal del procesador, ***IACK***, la lógica del dispositivo externo coloca su código de identificación (en nuestro ejemplo una señal de 8 bits) en el bus de datos del sistema. La ***UPC*** recibe los datos y los interpreta como un código de identificación del dispositivo que interrumpe; usa dicha identificación para armar la dirección donde se encuentra localizado el programa que atiende a la interrupción en cuestión.
4. Siguiendo un protocolo especificado por el sistema, el dispositivo externo coloca ahora sus datos a través del puerto de entrada y salida para que éste lo transmita al bus del sistema cuando se le solicite.

12.3.3 Prioridades de Interrupción

¿Qué pasa cuando más de un dispositivo interrumpe a la vez? Existen dos tipos de soluciones a este problema:

1. Se puede agregar lógica a la ***UPC*** que nos permita tener varias líneas de petición de interrupción cada una de ellas con una prioridad distinta a la otra.
2. Utilizar un circuito externo a la ***UPC*** que resuelva dichas prioridades y como salida tenga una sola línea de petición de interrupción.

Si las señales de interrupción son recibidas al mismo tiempo, se atiende primero a la que tiene la mayor prioridad, quedando los otros dispositivos pendientes de recibir atención. Los otros dispositivos pueden remover su señal de interrupción o si la mantienen, se atenderán uno a la vez, secuencialmente, en el tiempo como se indica en la figura *12.7a*.

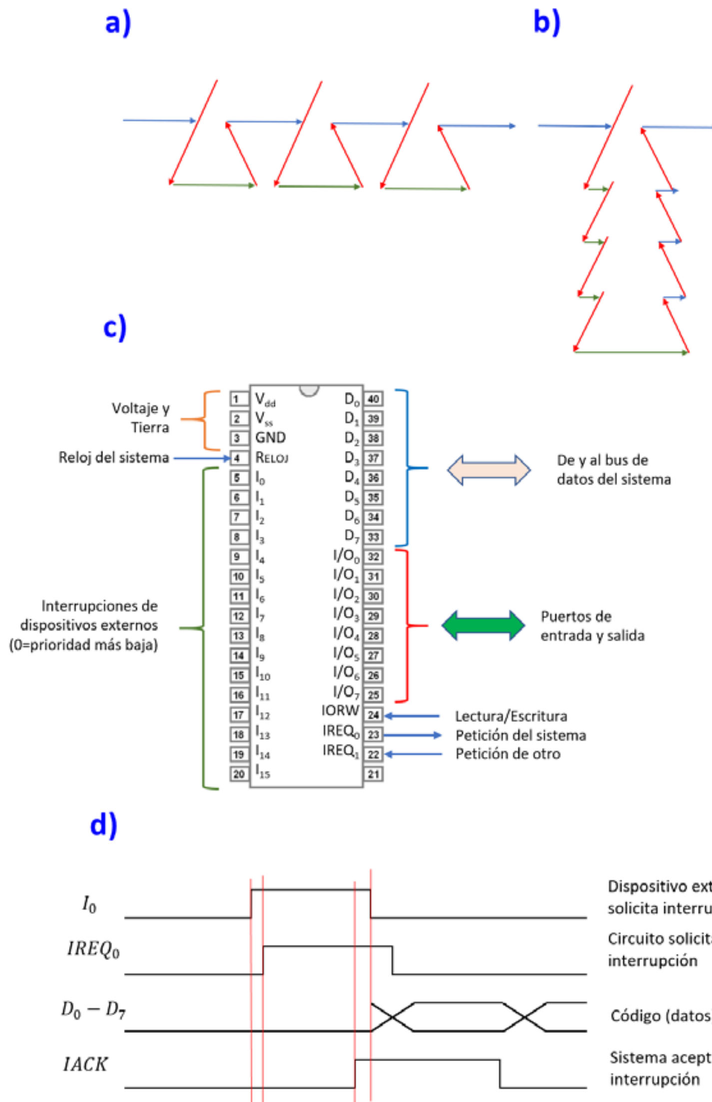


Figura 12.125 Prioridades en las interrupciones.

Si, por el contrario, las señales no llegan simultáneamente, las interrupciones serán procesadas según van llegando y puede llegar a formarse una cola de interrupciones pendientes de terminar que pudieran a veces llegan a saturar a la ***UPC***.

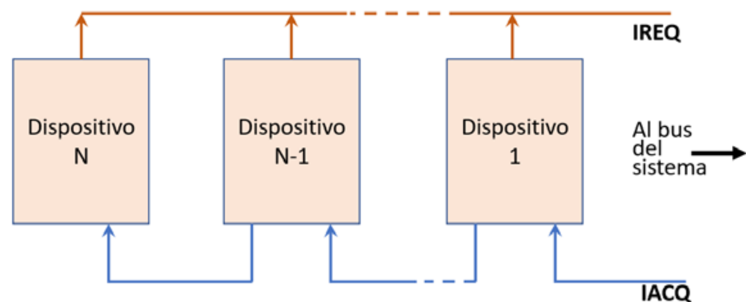
Si en cierto momento estamos atendiendo a un dispositivo de baja prioridad y somos interrumpidos por uno de mayor prioridad, la ***UPC*** guarda los registros (o nos permite hacerlo) y atiende a la nueva interrupción. Si llega otra señal de interrupción de más prioridad vuelve a suceder lo mismo (figura *12.7b*). Muchas veces es necesario atender sólo a una interrupción y no permitir otras

interrupciones hasta que no termine la actividad crítica. Las computadoras proveen los mecanismos para deshabilitar las interrupciones indeseadas y a esta acción se le conoce como ***enmascarar las interrupciones***, esto es, las interrupciones tienen que pasar por un tamiz antes de ser atendidas.

Los dispositivos para resolver las prioridades colocan en las líneas de datos un código de identificación que permiten a la ***UPC*** saber qué dispositivo obtuvo el permiso de transmitir o recibir (figura *12.7c* y *12.7d*).

Uno de los esquemas utilizados para resolver prioridades externas es la ***cadena de prioridades*** (Daisy Chain) en el que la señal de aceptación de interrupción llega al primero de los dispositivos; éste tiene que tomar la señal o pasarla al siguiente circuito. La desventaja de usar este esquema es que el último de la cadena recibe poca atención.

Figura 12.126 Interrupción por prioridad en cadena.



Existe otro tipo de interrupciones que exigen el máximo de atención y no pueden esperar. A este tipo de interrupciones con tan alta prioridad se les conoce como “no enmascarable” (non-mascarable Interrupt: ***NMI***) y no pueden ser deshabilitados. Generalmente se reserva esta línea de muy alta prioridad a eventos tales como falla inminente de energía o un error fatal del sistema que impide que éste pueda seguir trabajando, tal como un error de memoria.

Una falla de energía eléctrica puede ser detectada unos cuantos milisegundos antes de que no se pueda hacer nada. Unos pocos milisegundos son suficientes para que la mayoría de las máquinas puedan apagarse ordenadamente procurando hacer el menor caos posible en el sistema.

En la figura *12.9* presentamos el dispositivo para resolver prioridades conectado a las líneas del bus del sistema.

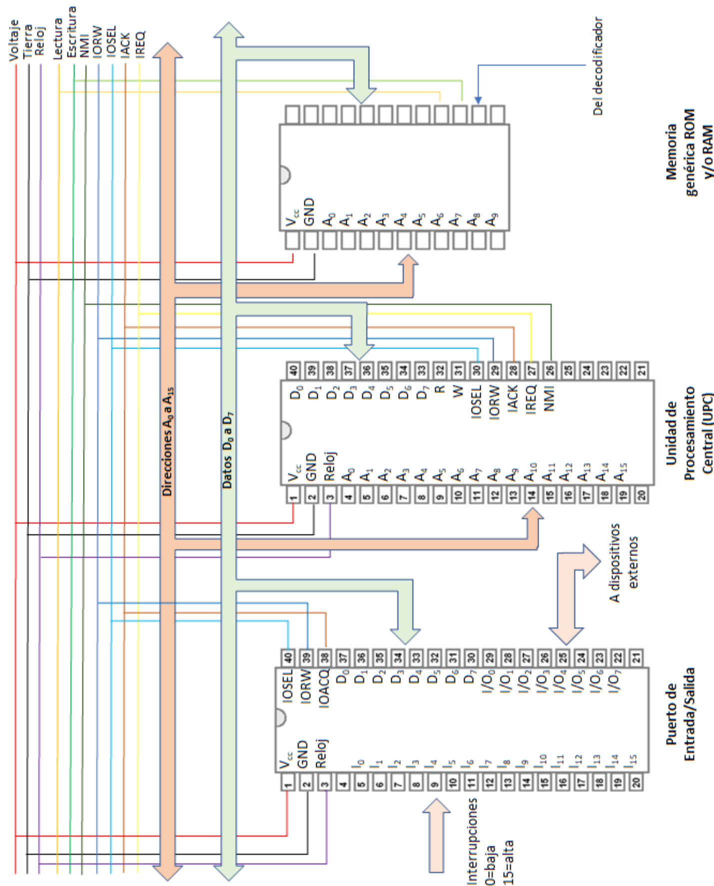


Figura 12.127 Sistema con prioridad de interrupciones.

12.4 Acceso Directo a Memoria

Como ya se observó en las secciones precedentes, el objetivo de la mayoría de las operaciones de entrada y salida es la transferencia de datos entre los dispositivos externos y la ***UPC***. Ya sea la ***E/S*** programada o la de interrupciones, se requiere de la participación activa de la ***UPC***, esto es, la ejecución de instrucciones para leer o escribir tanto a los dispositivos externos como a la memoria interna (principal). Esto puede no ser tan rápido como se piensa para ciertos procesos que requieren de extremada velocidad, como es el caso de la transferencia del contenido de memoria a la pantalla de visualización (***CRT***) o de y hacia la memoria de almacenamiento masivo, como son las memorias externas[40] de un sistema. Una vez más, la solución consiste en substituir los

[40] Almacenamientos secundarios y de respaldo.

programas por electrónica que realice la misma función, pero a más velocidad.

Una discusión completa del tema queda fuera del rango que pretende abarcar este libro por lo que, una vez más, referimos al lector interesado a la bibliografía para más detalles del tema.

Antes de decidir si es importante o no que la Unidad de Procesamiento Central intervenga en cada lectura o escritura de un dispositivo externo, debemos responder a la pregunta:

> ¿Es esta operación común o es un caso especial
> y aislado?

La respuesta es que este tipo de operaciones es la más habitual en un medio de computación y, de hecho, no sólo la ***UPC*** emplea muchos recursos y tiempo en leer datos de un dispositivo externo, sino que emplea tanto tiempo o más transmitiendo rutinariamente información de la memoria principal a los dispositivos externos.

Si la ***UPC*** no interviniera eventualmente en las transferencias de datos a dispositivos externos ¿en que podría ocupar su tiempo?

Solamente en las aplicaciones más sencillas y triviales la respuesta sería: En nada. Pero, claramente, mientras las aplicaciones se vuelven mucho más complejas, la pérdida de tiempo se empieza a convertir en un serio problema.

Podemos concluir entonces, sin temor a equivocarnos, que existen un gran número de aplicaciones en los que el tiempo perdido en atender todo tipo de interrupciones sería intolerable.

12.4.1 Robo de Ciclos

El ***acceso directo a memoria*** (Direct Memory Access; ***DMA***) provee la solución a las preguntas planteadas en la sección anterior. Crearemos un nuevo dispositivo para nuestro sistema de cómputo que se dedique exclusivamente a la transferencia de datos entre la memoria principal y los dispositivos externos.

Este dispositivo realiza sus operaciones pasando encima de las señales de la ***UPC*** y creando sus propias señales para el bus del sistema que habilite la transferencia de datos de un sitio a otro.

Una de las formas de deshabilitar la ***UPC*** consiste en detener su reloj temporalmente "robando" ciclos para otras actividades como se muestra en la figura *12.10*. En este circuito, el ***DMA*** roba ciclos de reloj a la ***UPC*** por medio de una señal de ***INHIBIR*** (*INHIBIT* o *HOLD*) que debemos agregar a nuestro bus del sistema.

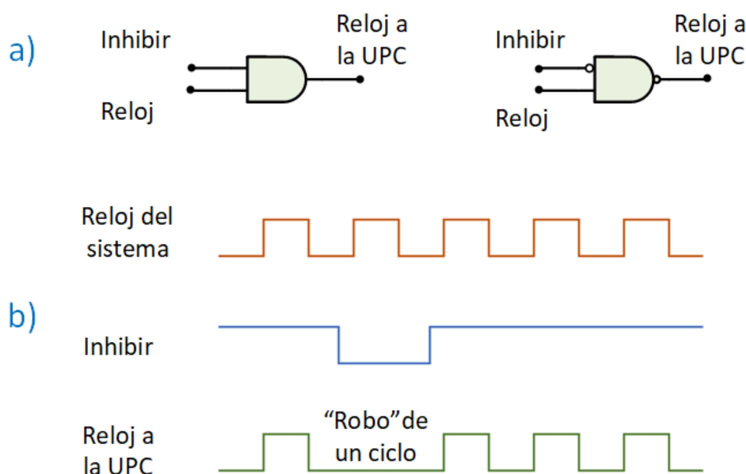


Figura 12.128 Robando ciclos al reloj.

Los dispositivos no tienen forma de saber de dónde provienen las señales del bus, por lo que el ***DMA*** puede tomar control completo del sistema mientras la Unidad de Procesamiento Central "duerme".

Si recordamos en detalle qué es lo que se necesita para una transferencia de datos, podemos deducir de qué elementos debe constar el ***DMA*** para poder lograr su propósito. Analizando la figura *12.11* encontramos, entre otros, los siguientes componentes:

1. Un Registro de Dirección que contiene la dirección de la siguiente palabra de memoria a acceder para una operación de lectura o escritura.
2. Un Contador que inicialmente contiene la longitud del buffer de datos que debe llenarse durante la operación de lectura o del que se leerán los datos en una operación de escritura.
3. Un Registro de Estado que identifica la dirección en que fluyen los datos, si el ***DMA*** está inactivo o activo y varias otras opciones.
4. La lógica y electrónica necesaria para sumar o restar las unidades necesarias a los registros según se requiera.

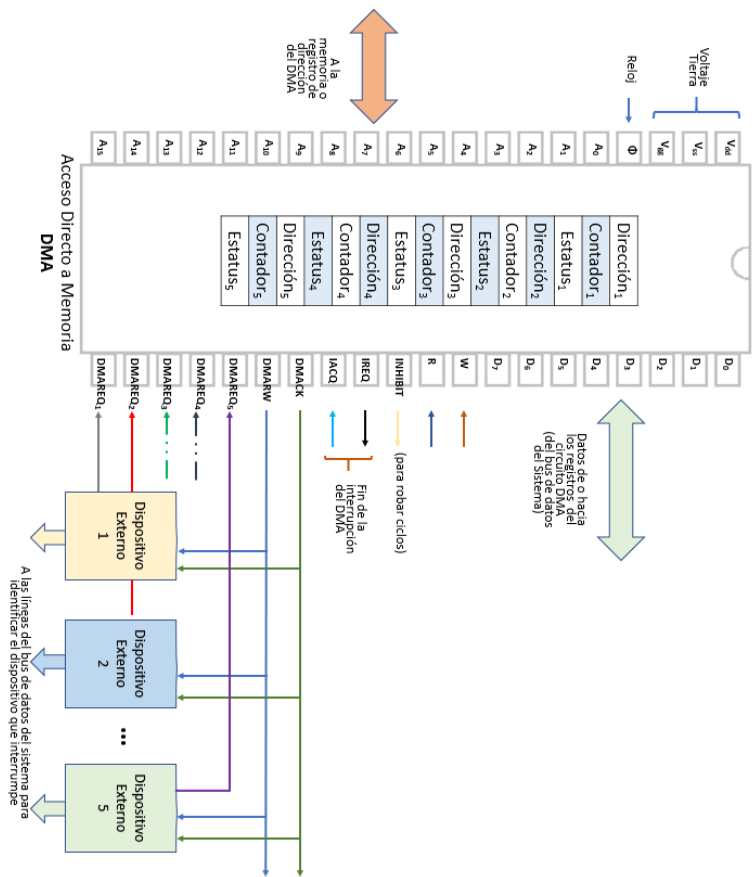


Figura 12.129 Acceso Directo a Memoria (DMA) de cinco canales.

Un programa en la ***UPC*** debe ejecutarse en un principio para cargar los parámetros que requiere el ***DMA*** para poder comenzar a funcionar. La ***UPC*** puede, en cualquier momento, preguntar el estado de las operaciones del ***DMA***. Un programa puede ajustar sus parámetros de acuerdo con qué tanto ha avanzado el ***DMA***; a esta operación se le conoce como ***pescar al DMA al vuelo.***

Un programa puede, también, en cualquier momento suspender o dar por terminadas las operaciones del ***DMA*** con sólo escribir una nueva palabra de estado en el Registro de Estado del ***DMA***.

Para evitar la interferencia de la ***UPC*** en las señales que viajan por el bus del sistema, se emplea una nueva lógica llamada de tres estados[41] (tri-state) en la cual, además de los dos ya conocidos en lógica digital, aparece un nuevo estado indistinto (deshabilitado).

[41] Estado de alta impedancia que da como resultado que la línea o conexión de un circuito aparezca desconectada eliminando su influencia en el resto del circuito.

A la operación de colocar las conexiones externas de la ***UPC*** en el tercer estado neutro se le conoce como flotar los buses.

Una vez que el registro de cuenta del ***DMA*** llega a cero, éste suspende su operación en espera de nuevas instrucciones.

12.4.2 DMA con Dispositivos Externos Múltiples

Un dispositivo ***DMA*** puede controlar las operaciones para muchos dispositivos externos. Puesto que los dispositivos ***DMA*** no realizan la transferencia de datos, no requieren de puertos de ***E/S*** a través de los cuales los dispositivos externos comuniquen con el sistema de la computadora. Tales dispositivos externos conectan directamente a través del bus externo del sistema. Este arreglo es muy conveniente puesto que permite que el dispositivo ***DMA*** controle el acceso ***DMA*** para un número definido de dispositivos externos. La figura *12.11* ilustra sólo una posibilidad.

En la figura *12.11* notamos que existen 5 dispositivos conectados al ***DMA***, A cada una de estas conexiones se le conoce como canales y en este caso estamos hablando de un ***DMA*** de 5 canales. Nótese que debe existir un juego de registros por cada canal que tiene el ***DMA***. Así mismo, existe una línea de petición de interrupción (***DMAREQ***) por cada uno de los canales del ***DMA***. Sin embargo, se usa una línea común para el reconocimiento de la señal del ***DMA*** (***DMACK***) y de la lectura y escritura del ***DMA*** (***DMARW***).

El dispositivo que interrumpe debe, a su vez, contener la lógica para su selección y ser el único que responda a ***DMACQ***. En este caso es la responsabilidad del dispositivo externo, y no la del ***DMA***, asegurar que sólo uno de ellos se considere seleccionado a la vez.

12.4.3 DMA Simultáneo

Observe que un ***DMA*** que roba ciclos reproduce la lógica de un ***CPU*** de forma limitada. Un circuito de Acceso Directo a Memoria puede igualarse a una ***UPC*** que sólo realiza dos funciones:

1. Transferir datos de un dispositivo externo a la memoria principal.
2. Transferir datos de la memoria principal a un dispositivo externo.

Ambas funciones deben realizarse a la máxima velocidad posible y en armonía con el sistema para no interferir con éste.

Debido a este limitado número de operaciones podemos eliminar algunas de las funciones que en una ***UCP*** consumen mucho tiempo, por ejemplo, traer instrucciones de memoria e interpretarlas aquí, no es necesario.

Podemos llevar la lógica del ***DMA*** un paso más allá duplicando todas las señales del bus del sistema en otro bus que llamaremos bus del DMA. De esta forma podemos eliminar el robo de ciclos de reloj de la ***UPC*** y se puede actuar cuando la ***UPC*** no esté usando los dispositivos externos o la memoria principal. Esto sucede en la parte alta del reloj, por lo que muy bien pueden utilizarse estos periodos para realizar las transferencias de datos.

Una vez más, las conexiones de los módulos de memoria y de otros dispositivos que estén conectados al bus del sistema deben actuar en lógica de tres estados para poder desconectarse de uno u otro bus, según se requiera.

Los ***DMA*** requieren del servicio de la ***UPC*** para inicializar los distintos registros necesarios para su operación, pero podemos tratar de diseñar un ***DMA*** que no requiera de la ***UPC*** y pueda cargar sus propios registros. Si agregamos algunas instrucciones de E/S tenemos en este punto un procesador especializado capaz de ejecutar de forma independiente las transferencias de datos entre memoria y dispositivos externos. A tal dispositivo se le conoce con el nombre de procesador de Entrada/Salida (I/O processor; ***IOPs***).

12.5 Entrada/Salida Serial

Muchos de los dispositivos externos requieren de una salida en serie en lugar de la paralela que hemos analizado hasta el momento. Entre estos dispositivos están[42] (lista no exhaustiva):

- Cinta magnética
- Teletipos
- Discos flexibles y duros (magnéticos o de estado sólido)
- Comunicaciones telefónicas vía Módem
- Impresoras
- Ratón
- Tarjetas de red local

[42] Muchos de ellos ya en desuso.

- Teclados
- Lectores de código de barras
- Escáneres

En capítulos anteriores hemos visto que un registro de corrimiento puede formar la base de una conversión serial-paralelo y viceversa; pero en aplicaciones más complejas se prefiere utilizar un circuito especializado para realizar todas las conversiones y cumplir con los estándares de comunicación en serie (como el estándar ***RS-232***).

Aunque en un principio se utilizó la transferencia en paralelo hacia la memoria secundaria, principalmente en sistemas de almacenamiento secundarios como discos duros internos. Muy pronto la evolución en la velocidad de las ***UPC*** saturó las líneas causando distintos problemas secundarios:

- Retraso de las señales.
- Interferencias electromagnéticas (***EMI***).
- Integridad de los datos.

La solución a todo esto llegó con el diseño de una nueva interfaz serial de alta velocidad compatible con la anterior paralela con menos conductores y trabajando a un voltaje reducido. Otra de sus ventajas es que los nuevos estándares incorporan su compatibilidad con dispositivos externos tipo ***USB*** (descritos más adelante en este capítulo).

Uno de los circuitos que en un principio cumple con ese propósito es el ***transmisor-receptor universal asíncrono*** o ***UART*** (Universal Asynchronous Receiver-Transmitter). El circuito es aún hoy en día usado para transmisiones en red donde no son accesible las líneas digitales ópticas de alta velocidad o para el envío de faxes donde así se requiere.

Otra forma de conversión depende enteramente de una solución por programa y consiste en leer un bit a la vez del puerto de ***E/S***. Llamaremos a esta modalidad ***Entrada/Salida Programada*** o ***PIO*** (Programmed Input/Output, Programmed I/O) y el ***CPU*** debe intervenir en cada transacción. Esto presenta las siguientes características:

- La ventaja del uso de un programa dentro de la ***UPC*** radica en su sencillez y la eliminación de todo circuito electrónico

externo que no sea indispensable. El costo total es reducido.

- Debido a que la ***UPC*** interviene a lo largo de todo el proceso, la solución es lenta y si se requiere de retrasos exactos para armar la palabra en memoria, no podemos lograrlo en un sistema con interrupciones.

El circuito ***UART*** tiene dos funciones específicas:

- Tomar una palabra en paralelo y convertirla en su equivalente serial agregando verificación de paridad y bits de arranque y parada.
- Tomar una entrada serial, verificar la paridad, quitar los bits de arranque y parada y entregar a su salida una palabra en paralelo de ***n*** bits (usualmente 8).

Un diagrama de un ***UART*** funcional aparece en la figura *12.12*.

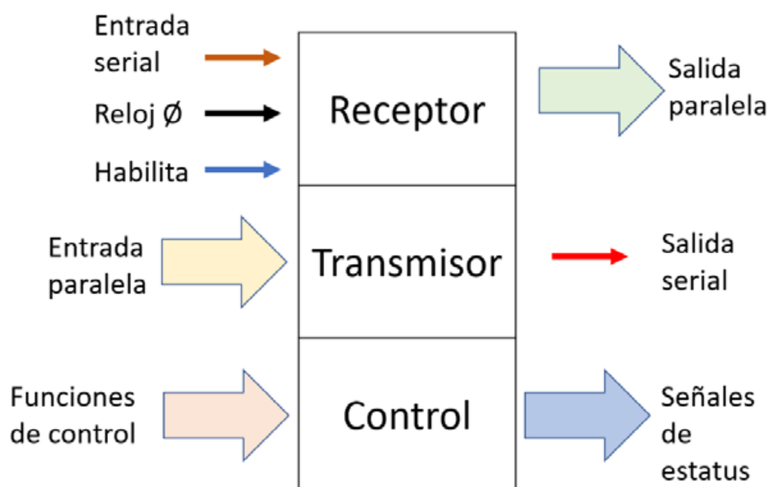


Figura 12.130 Diagrama de bloques de un UART.

Todos los ***UART*** tienen 3 secciones: un transmisor, un receptor y una sección de control.

Un ***UART*** requiere de un puerto de entrada y uno de salida para hacer la interfaz con el sistema de computación. Los ***UART*** modernos se han fabricado de tal forma que son directamente compatibles con el bus del sistema, ahorrándonos el puerto de ***E/S***. En la figura *12.3* mostramos un ejemplo de un ***UART*** que algunos fabricantes llaman ***interfaz adaptadora de comunicaciones asíncronas*** o ***ACIA***.

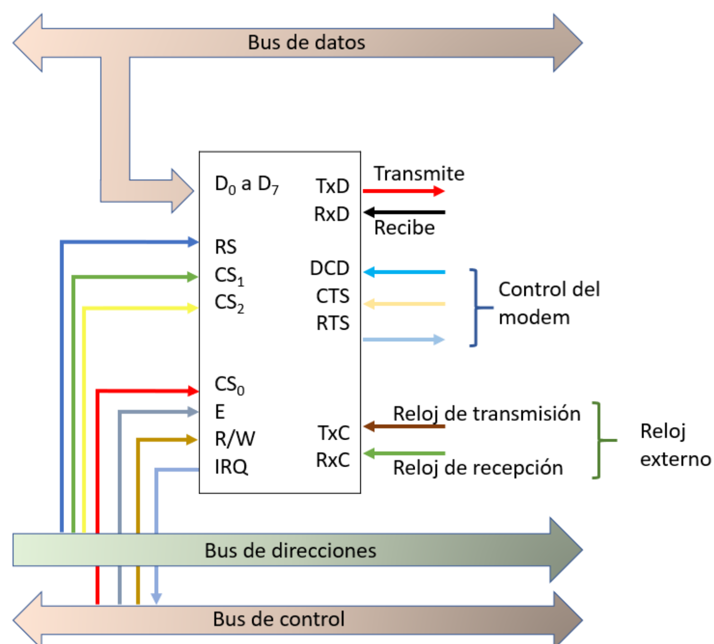


Figura 12.131 Ejemplo de un ACIA comercial (Motorola 6850).

Un circuito más avanzado, llamado ***USART***, responde también de forma síncrona y permite la conexión directa a un dispositivo conocido como ***Módem*** que monta (modula) la señal digital en una analógica para su transmisión por las líneas telefónicas y del otro extremo recibe la señal desmontándola de su portadora analógica (demodulación) (su nombre proviene de la contracción de las dos acciones que realiza MOdular y DEModular). Para esta interfaz se requieren de otras líneas de control que detectan varios estados del módem. En la figura *12.14* mostramos un diagrama funcional de un ***USART***.

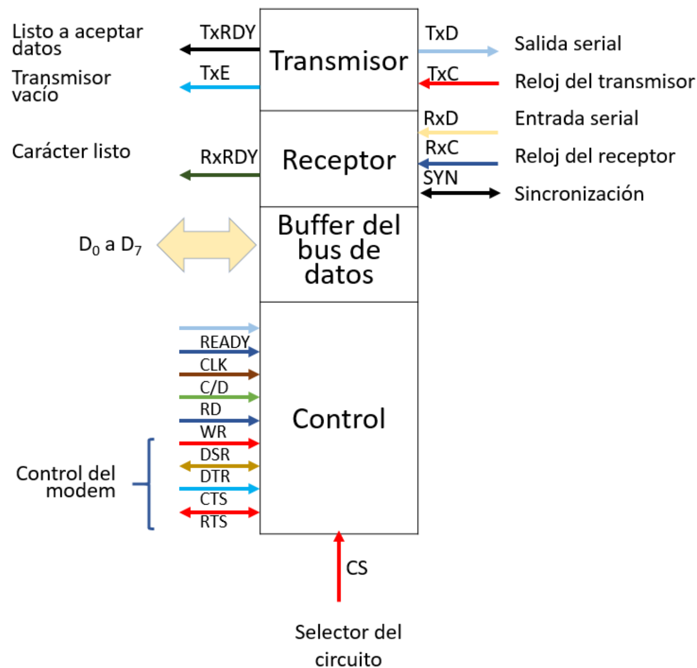


Figura 12.132 Diagrama de bloques de un USART (Intel 8251).

Entre las aplicaciones de estos circuitos podemos citar:

- Comunicaciones entre un *PC* y *PDA* (Personal Digital Assistant; Asistente Personal Digital).
- Transmisiones de audio, vídeo, juegos por infrarrojos o Bluetooth[43] (teléfonos móviles).
- Transmisión a paneles de matrices de diodos para despliegue de información, vallas publicitarias, etc.
- Módems
- Faxes
- Etc.

12.6 Estándar USB

Aunque el propósito principal de un computador es procesar datos ordenándolos, clasificándolos y aplicándoles algoritmos que los transforman de una forma u otra, su función se ve de gran forma disminuida si no se puede dejar una traza de su actividad y guardar dichos datos. Se requiere, entonces una interfaz entre la computadora y el mundo físico, donde almacenar dichos datos y, eventualmente presentarlos en una forma inteligible a los humanos, sus usuarios finales. A través de los años los estándares que

[43] Norma de comunicación que permite el intercambio bidireccional de datos a corta distancia utilizando ondas de radio *UHF* en una banda de frecuencia de 2.4 GHz.

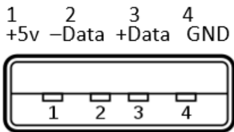
permiten conectar distintos dispositivos para intercomunicar computadoras o periféricos a sus salidas se han sucedido los unos a los otros:

- Interfaz paralela externa – un puerto, generalmente de salida, aunque en sus últimos años de vida fue modificado para ser también de entrada, que permitía conectar distintos periféricos a la computadora. Se conoció como puerto Centronics y luego como DB-25. La conexión se realizaba por medio de un cable en el que las señales viajaban todas juntas (los 8 bits requeridos que forman un byte) a velocidades entre 150Kb/s a 1Mb/s. Aunque diseñado originalmente sólo para impresoras, su posterior transformación a bidireccional lo hizo práctico para su uso con escáneres (scanner), unidades de almacenamiento secundario (ZIP) y algunos discos duros a distancias no mayores de unos cuantos metros.
- Puerto serial – a gran diferencia del puerto paralelo, aquí los datos viajan uno a uno (bit tras bit) transmitidos por un cable único junto con otros cables auxiliares para las señales de control. Este puerto fue diseñado desde un principio como bidireccional usando un conector DB-9. Fue usado en aplicaciones en donde, o la velocidad de transmisión no tenía gran importancia (por ejemplo, un teclado), o no había más remedio que transmitir de esa forma (Módem). Existía en velocidades desde 75 bits/s llegando hasta 256Kb/s. Su evolución desemboca en la interfaz ***USB*** que palía la complicada configuración técnica por la cual tiene obligatoriamente que pasar el usuario para su puesta en marcha.
- Puerto ***USB***. Analizado más adelante en esta sección

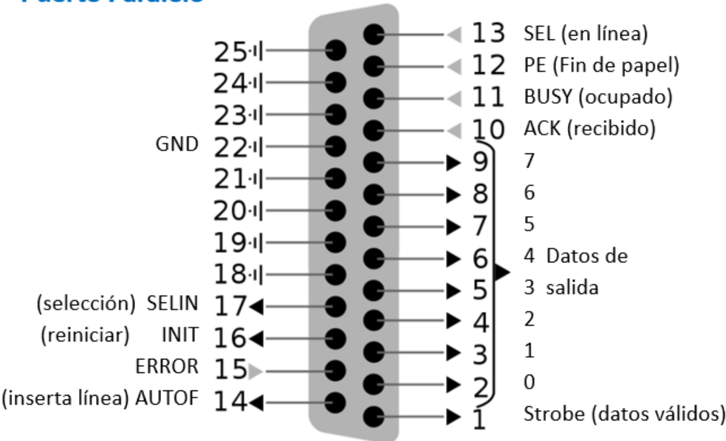
En la figura *12.15* puede ver los distintos conectores de cada una de las interfaces y el uso de sus patillas.

Figura 12.133 Conectores de distintos puertos externos.

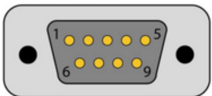
Puerto USB



Puerto Paralelo



Puerto Serial



1	DCD	Detección de portadora de datos
2	RX	Recibe datos
3	TX	Transmite datos
4	DTR	Datos listos
5	GND	Tierra
6	DSR	Conjunto de datos listos
7	RTS	Petición de envío
8	CTS	Limpia para enviar
9	RI	Indicador de campanilla

En 1994 un grupo de siete empresas; Compaq, DEC, IBM, Intel, Microsoft, NEC y Nortel se reunieron para iniciar el desarrollo de un nuevo estándar. El objetivo era facilitar fundamentalmente la conexión de dispositivos externos a las PC reemplazando la multitud de conectores en la parte posterior de las mismas. Se intentaba abordar los problemas de uso de las interfaces existentes y simplificar la configuración del software de todos los dispositivos conectados. Se requería también permitir una mayor velocidad de transmisión de datos para los dispositivos externos usando la sencillez de la interfaz serial existente. En 1995 nació el estándar ***USB***.

Antes de la llegada del puerto serial ***USB***, la interfaz paralela fue adaptada para acceder a una serie de dispositivos periféricos distintos de las impresoras. Uno de los primeros usos del puerto paralelo en su forma bidireccional fue para insertar una llave de hardware (dongles) que se suministraba con una aplicación y se usaba para proteger el programa contra su uso ilegal o copia. Otros usos incluyeron unidades de disco óptico tales como lectoras y grabadoras de ***CD***, unidades Zip (cartuchos magnéticos de alta capacidad usados para respaldo), escáneres, módems externos, interfaces de juego y joysticks. Algunos de los primeros reproductores de MP3 portátiles requerían una conexión de puerto paralelo para transferir material al dispositivo. Existían adaptadores disponibles para ejecutar dispositivos ***SCSI*** en paralelo. Otros dispositivos, como programadores ***EPROM*** y controladores de hardware, podían conectarse a través del puerto paralelo. Desde la perspectiva del usuario de la computadora, la interfaz ***USB*** mejora la facilidad de uso de varias formas:

- La interfaz ***USB*** se autoconfigura, eliminando la necesidad de que el usuario ajuste los parámetros del dispositivo tales como velocidad, formato de datos, interrupciones, direcciones de entrada / salida o canales de acceso directo a la memoria.
- Los conectores ***USB*** están estandarizados, por lo que cualquier periférico puede usar la mayoría de los receptáculos disponibles.
- ***USB*** aprovecha al máximo la potencia de procesamiento adicional que se puede poner económicamente en los dispositivos periféricos para que estos se autoadministren. Como tal, los dispositivos ***USB*** a menudo no tienen configuraciones de interfaz que el usuario pueda ajustar.
- La interfaz ***USB*** es intercambiable en caliente (los dispositivos se pueden intercambiar sin apagar o reiniciar la computadora huésped).
- Los dispositivos pequeños se pueden alimentar directamente desde la interfaz ***USB***, eliminando la necesidad de cables de alimentación adicionales.
- Debido a que el uso del logotipo de ***USB*** sólo se permite después de pruebas exhaustivas, el usuario puede confiar en que un dispositivo ***USB*** funcionará sin que tenga que interactuar con sus ajustes y su configuración.

- La interfaz ***USB*** define protocolos para la recuperación de errores comunes, mejorando la confiabilidad sobre otros tipos de interfaces usadas anteriormente.
- La instalación de un dispositivo basado en el estándar ***USB*** requiere una acción mínima del operador. Cuando un usuario conecta un dispositivo a un puerto ***USB*** de una computadora encendida y operando, ésta puede (prefiriéndose las opciones en el orden siguiente):
  - Configurarlo de forma totalmente automática utilizando los controladores de dispositivo existentes.
  - Descargarlos sin intervención del usuario y elegir el que más convenga o uno genérico.
  - Solicitar al usuario que busque un controlador, que luego instala y configura automáticamente.

El estándar ***USB*** también proporciona varios beneficios para los fabricantes de hardware y desarrolladores de software, específicamente en la relativa facilidad de su implementación:

- Elimina el requisito de desarrollar interfaces patentadas para nuevos periféricos.
- La amplia gama de velocidades de transferencia disponible se adapta a dispositivos que van desde teclados y ratones pasando por escáneres, impresoras hasta interfaces de transmisión de video.
- Se puede diseñar una interfaz ***USB*** para proporcionar la mejor latencia disponible para funciones de tiempo crítico o se puede configurar para realizar transferencias de datos masivos en segundo plano con poco o muy bajo impacto en los recursos del sistema.
- El diseño de la interfaz ***USB*** se realizó de forma generalizada sin usar líneas de señal dedicadas a una sola función de un dispositivo.

Como ocurre con todos los estándares, ***USB*** posee múltiples limitaciones en su diseño:

- Los cables ***USB*** tienen una longitud limitada, ya que el estándar fue diseñado para periféricos usados cerca de la computadora, no entre habitaciones o edificios. Sin embargo, se puede conectar un puerto ***USB*** a una puerta de enlace (gateway) que acceda a dispositivos distantes.
- Los puertos ***USB*** tienen una topología de red de árbol estricta y un protocolo maestro/esclavo para direccionar

dispositivos periféricos; esos dispositivos no pueden interactuar entre sí excepto a través del anfitrión; dos anfitriones no pueden comunicarse directamente a través de sus puertos ***USB***. Es posible ampliar esta limitación a través de diseños ***USB*** especiales (USB On-The-Go).

- Un anfitrión no puede transmitir señales simultáneamente a todos los periféricos; cada uno debe direccionarse individualmente. Algunos dispositivos periféricos de muy alta velocidad requieren velocidades constantes no disponibles en el estándar ***USB***.
- Si bien existen convertidores entre ciertas interfaces heredadas y el ***USB***, es posible que éstas no proporcionen una implementación completa. Por ejemplo, un convertidor de puerto ***USB*** a puerto paralelo puede funcionar bien con cierta impresora, pero no con un escáner que requiera el uso bidireccional de las líneas de datos.
- Para un desarrollador de productos, el uso de ***USB*** requiere la implementación de un protocolo complejo e implica un controlador "inteligente" en el dispositivo periférico. Los desarrolladores de dispositivos ***USB*** destinados a la venta al público generalmente deben obtener un ID para su periférico ***USB***, lo cual requiere pagar una tarifa al "Foro de Implementadores ***USB***". Los desarrolladores de productos que utilizan la especificación ***USB*** deben firmar un acuerdo con dicho Foro. El uso del logotipo ***USB*** en su producto requiere pagar cuotas anuales y membresía a la organización.

Enumeramos las distintas características de cada puerto en la siguiente tabla (tabla *12.1*).

Tabla 12.1 Comparación entre distintos puertos de E/S			
Característica	**USB**	**Serial**	**Paralelo**
Ancho de banda	480 Mbps a 40 Gbps	Hasta 256Kbps	115KBps a 3MBps (EPP/ECP)
Numero de Dispositivos	127 dispositivos en un solo bus	Limitado al número de buses en la computadora	Limitado al número de puertos en la computadora
Corriente disponible en el conector	Sí, desde 500mA @ 5V hasta 900mA @ 5V	No	No
Límite de longitud del cable	Cualquiera mientras se cumpla la especificación (en general entre 1 y 5mts)	3m	1.8m
Plug'n'Play (conectar y utilizar) ***PnP***[44]	Si	No	No
Intercambiar en "caliente" (Hot Swappable)[45]	Sí	No	No

12.7 Resumen

Hemos dicho que un sistema de cómputo sólo debe contar con la ***UPC*** y la memoria principal. Aunque esto es estrictamente cierto, la funcionalidad del circuito sería casi nula. Es necesario, entonces, agregar toda una serie de circuitos y dispositivos que hagan que el diseño sea funcional y práctico para el que lo usa.

La interfaz a estos dispositivos y circuitos se realiza por medio de los puertos de Entrada/Salida. Es por estos puertos que la computadora se comunica y recibe información del mundo externo.

[44] PnP es una tecnología que permite reconocer rápida y automáticamente los periféricos compatibles durante su conexión sin necesidad de reiniciar la computadora. La intervención del usuario es mínima reduciendo errores de manipulación y configuración

[45] El intercambio en caliente es el reemplazo o la adición de componentes a un sistema informático sin detener, apagar o reiniciar el sistema. Opuesto a Cold-Swappable o intercambio en frio (con el sistema apagado).

12.7.1 Puntos Importantes del Capítulo

- La transferencia de datos entre el sistema de cómputo y el mundo externo se le conoce como Entrada y Salida (***E/S*** o ***I/O***).
- La ***E/S*** se puede dividir en Programada, por Interrupciones y Acceso Directo a Memoria (***DMA***).
- La ***E/S*** programada es conocida también como ***E/S*** de mapa de memoria pues se divide el mapa de memoria entre memoria y dispositivos de ***E/S***.
- Para evitar dividir el mapa de memoria se prefiere agregar más líneas de control a la ***UPC*** que seleccionen entre la memoria y los dispositivos de ***E/S***.
- La ***E/S*** por Interrupción solicita atención al sistema por medio de una línea.
- Para lograr la ***E/S*** por Interrupciones, se agregan varias líneas de control a la ***UPC***.
- Si la ***UPC*** es interrumpida debe realizar una secuencia ordenada antes de atender a la interrupción y esto incluye, entre otras cosas, el preservar el contenido de los registros.
- El dispositivo externo que interrumpe debe proporcionar un código de dispositivo o un vector de interrupciones que es la dirección donde se encuentra el programa que atiende al dispositivo.
- Es importante tener prioridades en las interrupciones. Esto se resuelve con un circuito que atiende las peticiones de los dispositivos externos y asigna las prioridades.
- El ***DMA*** consiste en guardar o leer datos de la memoria principal sin la intervención de la ***UPC***. Para esto, es necesario generar todas las señales que requiere la memoria y que en condiciones normales genera la propia ***UPC***.
- El ***DMA*** se consigue robando ciclos al reloj de la ***UPC*** o deteniendo su proceso.
- Para que las señales no interfieran una con otra cuando se generan de dos fuentes distintas, se utiliza una nueva lógica llamada de tres estados donde uno de ellos es inactivo y los otros dos corresponden a la lógica binaria.
- Un caso especial de ***E/S*** es el serial que sirve para la comunicación en muchos casos. El dispositivo que se encarga de llevar el control de esta comunicación es llamado ***UART***.

- Ciertos puertos anacrónicos de comunicación con el mundo externo han sido substituidos o han evolucionado a estándares que pueden mantener el paso de las velocidades de transferencia en constante evolución de los procesadores modernos.

12.8 Problemas

12.1 Diseñe un sistema completo que conste de un dispositivo de puerto de ***E/S*** paralelo, memoria ***RAM*** y ***ROM*** y la ***UPC***, escoja para esto circuitos comerciales. No use las conexiones de los circuitos, sino un diagrama esquematizado.

12.2 Consiga el diagrama de algún circuito ***DMA***.

12.3 Agregue un ***DMA*** al circuito anterior.

12.4 Consiga los diagramas de algunos circuitos funcionales de puertos programables de ***E/S*** (*PIA*; Peripheral Interface Adapter).

12.5 Consiga los diagramas de algunos circuitos funcionales de *UART*, *USART* y *ACIA*.

13 Programación

No se debe pasar por alto que un sistema de computación, no importando la complejidad de los varios componentes electrónicos que lo formen, no es más que un juguete caro a menos de que exista un propósito de su existencia y un programa que haga que este propósito se cumpla. Es este programa, formado por una serie de instrucciones precisas y arregladas en un orden lógico, que hacen de la computadora una herramienta útil.

Existen muchos tipos de programas y de sistemas de programas, a los que en conjunto se conocen como ***Programas*** (o su término en inglés: software), en oposición a los elementos mecánicos y electrónicos llamados en conjunto ***Hardware*** (anglicismo). Algunos de estos programas se usan para una tarea en particular como puede ser el caso de una nómina. Otros programas se usan para manejar los distintos puertos de entrada y salida de una computadora. La operación tanto mecánica como lógica de los dispositivos externos al sistema por donde la información es introducida y entregada (ver el capítulo *15*), es controlada directamente por estos programas. Existe aún otro tipo de programas que realizan funciones muy específicas de computación como realizar las funciones que la ***UAL*** no puede realizar o realiza muy despacio. Algunas veces se les conoce a los programas como subrutinas pues forman parte de programas mucho más grandes. Para un sistema de computación dado, muchos programas pueden reunirse y guardarse en un sistema de almacenamiento externo tal como discos duros magnéticos (o de estado sólido), llaves ***USB*** (medio muy común) o sencillamente en respaldos desmaterializados; posteriormente se utiliza un programa que pueda extraer a los primeros del medio donde fueron almacenados y ejecutarlos. A este programa se le conoce como ***sistema operativo*** (operating system) y si se encuentra almacenado en un disco magnético, se le llama sistema operativo de disco (disk operating system o ***DOS***).

Bootstrap

De origen en el inglés en el siglo 19 en el que las botas tenían una lengüeta para usar un gancho que ayudaba a apretarlas. Su uso se extendió como metáfora para significar literalmente “levantarse a sí mismo” o ejemplo de una tarea imposible.
En el sentido informático la computadora “se levanta a sí misma”; es decir sin intervención externa y por sus propios esfuerzos.

La primera consideración sería ¿Cómo hacer que la computadora comience a funcionar? Aplicando la energía eléctrica únicamente no es suficiente para hacer que la computadora comience a trabajar como el usuario desea. Debe existir un programa que haga la inicialización del sistema y determine las metas a alcanzar para poder hacer que el sistema esté en posibilidad de aceptar entradas de los dispositivos externos, como pueden ser el teclado o los medios de almacenamiento. A este programa se le conoce como programa de arranque (start-up program). Usualmente se le conoce en inglés con el nombre de bootstrap o programa de arranque en frío (cold-start); los programas de arranque no necesariamente tienen que ser muy grandes. El procedimiento común consiste en colocar en una localidad de memoria predefinida del tipo ***ROM***, un pequeño programa de arranque. Toda computadora tiene una forma determinada de comenzar la ejecución de un programa en una localidad especifica al aplicarles la energía eléctrica, esta característica se aprovecha para guardar en esa localidad el inicio del programa de arranque.

El programa de arranque inicia una cadena de eventos que termina con la ejecución, en memoria principal, de una parte del sistema operativo conocida como ***procesador de comandos***. Éste queda en todo momento activo para recibir órdenes posteriores de otros programas o del usuario directamente en una parte del sistema operativo llamado ***BIOS*** o sistema básico de entrada y salida (Basic Input Output System).

Lo que sucede a partir del punto en que el sistema operativo toma control de la Unidad de Proceso Central, varía de un sistema a otro, pero el objetivo fundamental de este programa es coordinar las distintas operaciones del computador y llevar registro del estado de varios eventos tales como interrupciones, peticiones a dispositivos externos, etc. Una discusión a fondo de sistemas operativos queda fuera del alcance de este libro, pero se puede consultar la bibliografía para ahondar en el tema.

13.1 El Sistema Básico de Entrada y Salida (BIOS)

El ***BIOS*** (Sistema Básico de Entrada y Salida), como ya se explicó brevemente, es un programa conformado por una serie de instrucciones utilizadas para realizar la inicialización de todos los componentes de la computadora durante su proceso de arranque. Proporciona servicios de tiempo de ejecución para el sistema operativo y los programas del usuario que así lo requieran.

El microprograma (firmware) que forma el ***BIOS*** viene preinstalado en la placa del sistema y es el primer programa que el ***CPU*** ejecuta cuando se enciende.

El ***BIOS*** en las ***PC*** modernas inicializa y prueba los componentes de hardware del sistema en lo que se llama ***Auto prueba de inicio*** (***POST***; Power-On Self Test). Esta auto prueba garantiza que no falte ningún componente esencial en la computadora y que se reúna la funcionalidad necesaria para iniciarse con éxito, como la existencia de la memoria, un teclado y otros componentes básicos. Si se detectan errores durante la prueba, la computadora proporciona un código que revela el problema. Los códigos de error suelen presentarse como una serie de pitidos que se escuchan poco después del inicio.

Antiguamente el ***BIOS*** proporcionaba una capa de abstracción de hardware para el teclado, la pantalla y otros dispositivos de entrada/salida que estandarizaban una interfaz para los programas de aplicación y el sistema operativo. Los sistemas operativos más recientes no utilizan ya el ***BIOS*** después de la carga (bootstarp) inicial, sino que acceden directamente a los componentes electrónicos requeridos.

Todos los ***BIOS***, desde los más antiguos hasta los más modernos, proveen un mecanismo, primitivo o sofisticado, para que el usuario interactúe con él proporcionado ciertos parámetros, desde los más simples como por ejemplo la hora y fecha o los más complejos como el voltaje y velocidad del reloj del ***CPU***[46], que se guardan en una pequeña memoria protegida por una batería.

La mayoría de las implementaciones de ***BIOS*** están diseñadas específicamente para funcionar con un modelo de placa base o computadora en particular, al interactuar con varios dispositivos que componen el conjunto de chips del sistema complementario de un fabricante específico. Originalmente, el microprograma de la ***BIOS*** se almacenaba en un chip ***ROM*** en la placa base de la ***PC***. En los sistemas informáticos modernos, el contenido de la ***BIOS*** se almacena en una memoria flash (***EEPROM***) para poder reescribirla sin substituir el chip de la placa base. Esto permite actualizaciones sencillas del firmware del ***BIOS*** de parte del usuario final, agregando nuevas funciones o corrigiendo errores. Pero esto también

[46] La manipulación de ciertos parámetros puede dañar de forma permanente ciertos componentes de la placa madre (motherboard). Tal es el caso de los voltajes de alimentación o la velocidad del reloj (overclocking).

crea la posibilidad de que la computadora se infecte con programas maliciosos (rootkit) que afectan el ***BIOS***. Además de ello, una actualización fallida del ***BIOS*** puede bloquear el sistema de forma permanente, a menos que el sistema incluya algún tipo de copia de seguridad para este caso.

13.1.1 Interfaz de Microprograma Extensible Unificada (UEFI)

Una de las principales motivaciones para la creación de una nueva generación de ***BIOS*** llamada Interfaz de microprograma extensible unificada (Unified Extensible Firmware Interface; ***UEFI***) la constituyó el rápido desarrollo de la electrónica que pronto mostró las limitaciones de las antiguas ***BIOS*** creadas para un ambiente de 16 y 32 bits. En especial en lo concerniente a la limitación de 1MB de memoria de direccionamiento y discos duros que ya, para cuando se definía el estándar del ***UEFI***, lograban superar los 2TB de capacidad de almacenamiento.

Las ventajas de elegir una instalación de modo de arranque ***UEFI*** incluyen las siguientes:

- Evita las restricciones de dirección de ***ROM*** que limita el número de dispositivos externos conectados y residentes en la tarjeta madre (usualmente limitados a 128).
- Admite particiones de arranque del sistema operativo de más de 2 terabytes (2 TB) de tamaño.
- Integra la configuración de dispositivos ***PCIe***[47] en los menús de configuración del ***UEFI*** mismo.
- Incluye las imágenes de sistemas operativos ejecutables en una lista de arranque como entidades etiquetadas. Por ejemplo, la etiqueta del administrador de arranque de Windows en lugar de una etiqueta de un dispositivo sin procesar.

Los beneficios del modo de arranque ***UEFI*** sobre el modo de arranque ***BIOS*** incluyen:

- Se pueden usar particiones de disco duro de más de 2 TB.
- Reconoce más de cuatro particiones en una unidad de disco duro (divisiones lógicas para crear compartimientos diferentes dentro de un disco para que el usuario los perciba como si hubiera más de un disco duro).

---

[47] *PCI* express (Peripheral Component Interconnect Express) bus de expansión serial de alta velocidad usado para discos duros de estado sólido (*SSD*), tarjetas gráficas (*AGP*) y otros.

- Arranca mucho más rápido.
- Gestiona eficientemente el sistema y la energía.
- Es más fiable y gestiona mejor los fallos.

Pero también existen ciertas desventajas:

- En su arranque el ***UEFI*** verifica una llave única antes de proceder, esto evita la contaminación con virus, pero también puede hacer que un fabricante impida la instalación de cierto componente o programa (aún no sucede esto último, pero…).
- Es más complicado de entender para el usuario neófito.
- Es más difícil de configurar.
- Requiere una matriz de memoria de mayor capacidad en la placa madre (más cara).
- No es compatible con sistemas operativos muy antiguos (en simuladores o sistemas virtuales).
- No es compatible con ciertos hardware, pero generalmente, esto no es un problema.

13.2 Los Programas en la Memoria

Como hemos visto en la descripción de la Unidad de Procesamiento Central y en especial de la Unidad de Control, la computadora sólo es capaz de ejecutar una instrucción a la vez (en las arquitecturas tradicionales, no incluyendo en esto arquitecturas en paralelo) por lo que se requiere de una memoria donde guardar las instrucciones que forman el programa. La cantidad de memoria necesaria depende de la complejidad del programa y de la cantidad de datos que genere o use.

La ejecución del programa consiste en tomar una instrucción de la memoria principal (o varias según su arquitectura) e interpretarla en la Unidad de Control de la ***UPC***. El registro de Contador de Programa (***PC***) se utiliza para llevar un apuntador de memoria de la siguiente instrucción a ejecutar. Como ya hemos visto, el ***PC*** se debe incrementar un poco antes de realizar la transferencia (fetch) de la siguiente instrucción en memoria al Registro de Instrucciones (***I***).

Las instrucciones son utilizadas para especificar la secuencia lógica que debe ocurrir dentro del sistema de cómputo. Para usar una computadora debemos, por lo tanto, seleccionar primero los dispositivos que nos den la capacidad lógica suficiente y

posteriormente realizar la secuencia lógica que satisfaga nuestras necesidades. La secuencia lógica forma el programa y ***programar*** es crear estas secuencias.

Nada nos impide crear estas secuencias lógicas en forma binaria y de alguna forma colocarlas en la memoria en las localidades escogidas. De hecho, las primeras computadoras eran programadas de esta forma: una secuencia de 1 y 0 era cuidadosamente codificada y luego introducida a la máquina por medio de alambres que cambiaban el contenido de la memoria (en las computadoras más antiguas) o por medio de interruptores que se conectaban directamente a la memoria por medio del bus de datos (en la siguiente generación).

Los programadores rápidamente escogieron representar los códigos de instrucciones en hexadecimal en lugar de binario para abreviar y facilitar su labor (es muchas veces más fácil identificar un error en una secuencia de números hexadecimales que en una cadena de unos y ceros) y luego buscar la forma en que la máquina trabajara más ayudando a la conversión.

Bug

De origen en los EE.UU. en los años 40 el término se atribuye popularmente a la matemática, informática y contralmirante Grace Murray Hopper mientras trabajaba en una computadora Mark II. Ella y sus asociados descubrieron una polilla atascada en un relé que impedía la operación de la máquina por lo que comentó que estaba eliminando el “bug” (insecto) del sistema.

Al proceso de encontrar y corregir esos errores se le conoce con el nombre de ***espulgar*** o ***depurar*** el programa y a dichos errores algunas veces se les identifica o con el anglicismo ***bug*** usado como sustantivo o verbo.

Un código escrito en un papel u otro medio, forma lo que llamamos ***código fuente***, este código es introducido luego a la máquina y el programa que lo lee, lo convierte y coloca en su localidad de memoria es llamado un ***editor***. Es la forma más elemental de colocar los unos y ceros en donde deben de estar. A esos unos y ceros colocados en la memoria principal y listos a ser interpretados como un programa se les llama ***código objeto*** y no es más que la representación exacta de lo que en un principio teníamos en papel, pero ahora colocado en la memoria de la computadora.

Pronto se vio que el camino correcto consistía en hacer el trabajo del programador más fácil mientras que el de la máquina debería ser el más difícil. Simplificar el trabajo del programador es una mejora sustancial en nuestro equipo. Llevando este razonamiento un paso más allá, ¿Por qué no usar un lenguaje que sea más sencillo de aprender, programar e introducir a la máquina que una secuencia de números sin sentido más que para el más experto?

La respuesta a esto, en nivel sistema de computadora, se encuentra en el lenguaje ***ensamblador*** que no es más que convertir las instrucciones que entiende la Unidad de Proceso Central (lenguaje de máquina) en una serie de símbolos alfabéticos y números mucho más convenientes para los programadores humanos. Los lenguajes ensambladores son, en una primera aproximación, versiones simbólicas y mucho más fáciles de leer que los lenguajes de programación de máquinas computadoras. Al incrementarse el uso de las computadoras, los programas fueron cada vez más complejos y de mayor tamaño y se reconoció la necesidad de construirlos en unidades o módulos más pequeños e independientes. Estas características fueron pronto introducidas en los lenguajes ensambladores que permitieron que, módulos de programas, escritos como unidades independientes fuesen integrados o ensamblados en uno más grande; de ahí el nombre de ensamblador. En la época actual, casi todo tipo de computadora sea grande o pequeña, se vende con un lenguaje ensamblador y otros programas asociados que nos permiten introducir nuestras instrucciones a la memoria de la máquina. El lenguaje ensamblador es definido por el fabricante y es único para cada modelo de ***UPC***. Ocasionalmente, varios lenguajes ensambladores pueden existir para la misma máquina, pero el programa que se encarga de colocar los unos y ceros en la memoria, debe entregar el mismo resultado: el programa objeto ejecutable por la computadora.

Aunque algunos estándares para lenguaje ensamblador han sido propuestos, hay poca evidencia de que los fabricantes se apeguen a él, pero un patrón general ha emergido:

- Cada instrucción ocupa una línea.
- Cada línea se forma de cuatro campos:
    - Etiqueta
    - Mnemónico
    - Operando
    - Comentario
- Cada línea fuente de programa representa una línea objeto traducida por el programa que se denomina ensamblador.

El campo del mnemónico es el más importante y es el único campo obligatorio que aparece en cada línea.

```
etiqueta  mnemónico   operando  comentario
```

```
          LIM DC0, ADDR1   ; Carga dirección fuente en DC
AQUÍ:     LMA              ; Carga palabra en acumulador
          AIA H"0F"        ; Quita 4 bits de la palabra
          BZ SALIR         ; Salir de programa si
                           ; el resultado es cero
          SRA              ; Guarda bits modificados
          INC DC0          ; Incrementa el contador de datos
          JMP AQUI         ; Regresa a procesar siguiente byte
SALIR:                     ; Pasa a la siguiente instrucción
```

Nótese que lo único sagrado e inalterable son los códigos binarios de las instrucciones; los mnemónicos del ensamblador pueden ser alterados siempre y cuando se reprograme al programa ensamblador para reconocerlos y hacer la traducción adecuada.

El campo de las etiquetas puede o no tener información y si la contiene, sirve al ensamblador para llevar apuntadores de sitios interesantes dentro del programa a los que seguramente queremos regresar. Estos sitios son apuntadores en memoria pues no tenemos forma de usar la dirección verdadera al no saber en qué lugar la pondrá el ensamblador. La dirección en donde comienza a ensamblarse el programa depende de la longitud del sistema operativo, del tamaño del ensamblador, de instrucciones específicas, etc.

El campo de los operadores sirve para proveer información necesaria a la instrucción en caso de que exista. Por ejemplo, no podemos sumar un número al registro Acumulador si no indicamos en el campo de operando este número.

Los comentarios pueden o no incluirse, pero su uso es casi obligatorio pues contiene información que nos hace más fácil comprender qué es lo que hace el programa y, si es necesario, en su caso modificar o dar mantenimiento a programas escritos por otra persona o por nosotros mismos tiempo atrás. La información contenida en este campo es de vital importancia para el programador, pero no influye en nada al resultado del ensamblador.

13.3 Conjunto de Instrucciones

El grupo de instrucciones usados por una Unidad de Procesamiento Central particular es conocido como ***conjunto de instrucciones***. Cada fabricante incorpora un conjunto de instrucciones ligeramente distinto en cada ***UPC***, de acuerdo con el uso al que se destinará el producto. Algunos son muy sencillos consistiendo en un puñado de instrucciones, mientras que otros son bastante complejos, con su total llegando a los varios cientos. Existen varios

grupos de instrucciones, cada uno de ellos con un propósito específico tales como:

- de entrada y salida
- de transferencia
- condicionales
- aritméticas
- de interrupción
- etc.

Usualmente se les da un nombre que es una abreviatura en la forma de un grupo de caracteres, claro que, como ya sabemos, todo esto llega finalmente a ser unos y ceros almacenados en una localidad de la memoria principal.

13.4 Ensambladores

Un programa sencillo como el mostrado en la sección *13.2*, no puede ser ensamblado tal como se presentó; es necesario informar al programa ensamblador de varias cosas antes de que pueda comenzar a realizar su trabajo. Por ejemplo, es necesario informarle en qué localidad de memoria queremos comenzar a ensamblar el programa.

Existen una clase de instrucciones llamadas ***directivas del ensamblador*** que no forman parte del lenguaje de la máquina, pero proveen al ensamblador la información que no puede deducir por sí mismo.

Para poder armar las direcciones de las etiquetas que el programa contiene, es necesario especificar el origen del programa. Esto se logra con la directiva “origen” (***ORG***) que es la única directiva obligatoria y necesaria.

Para terminar el programa se usa la directiva “fin” (***END***) que informa al ensamblador que ya no siguen más instrucciones.

La directiva “iguala” (***EQU***) nos hace más fácil la labor de programar al asignar valores a variables para su uso posterior.

Otra de las directivas muy usadas es la asignación de una o dos palabras para asignar valores a variables (***DB*** y ***DW***) para su uso posterior.

Se debe recordar que las abreviaciones de las directivas del ensamblador cambian de uno a otro y siempre se debe consultar el manual. Existen ensambladores bastante complejos que permiten toda una serie de comandos complicados que hacen más sencillo el programar a nivel ensamblador.

Nuestro programa, usando las directivas explicadas, quedaría:

```
    ORG   H"03FF"
mascara   EQU H"0F"
    LIM   DC0,ADDR1    ; Carga dirección fuente en DC
AQUI:     LMA          ; Carga palabra en acumulador
    AIA   mascara      ; Quita 4 bits de la palabra
    BZ    SALIR        ; Sale de programa si el resultado es cero
    SRA                ; Guarda bits modificados
    INC DC0            ; Incrementa el contador de datos
    JMP AQUI           ; Regresa a procesar siguiente byte
SALIR:    END          ; Pasa a la siguiente instrucción
```

Una vez definido el inicio y fin de un programa, el ensamblador puede llevar a cabo su trabajo y nos entregará una secuencia de números hexadecimales que forman el programa objeto que correrá directamente en la máquina tal como se muestra a continuación (el programa se modificó ligeramente para ser ejecutado en un procesador del tipo 80x86 y se le agregaron todas las directivas del ensamblador necesarias para poder ensamblarlo):

```
Microsoft (R) Macro Assembler Version 14.26.28806.0 4/15/20
06:30:48
Page 1-1
;****************************************************************
; Programa anterior modificado ligeramente para un procesador
; tipo 80x86 con resultado de un ensamble.
; Nótense todas las directivas agregadas al ensamblador
;****************************************************************
; Definiciones previas usadas en el programa
000F      mascara     EQU 0Fh
; ---------------------------------------------------------------
0000      datarea SEGMENT ; Área definición dirección a convertir
0500      ORG 500h ; Dirección de inicio de la zona de datos
0500 ???? addr1 DW ?
0502      datarea ENDS
;*****************************************************************
0000      ejemplo SEGMENT  ; Programa ejemplo
;-----------------------------------------------------------------
--------------------------------
0000      codigo      PROC FAR
0100                  ORG 100h
ASSUME CS:ejemplo, DS:datarea, ES:ejemplo ; Donde está el código
0100      inicio:
0100 8D 1E 0500       LEA BX,addr1 ; Carga dir. fuente en reg.
                                   ; auxiliar
0104 89 07 AQUI:      MOV [BX],AX ; Mover datos de dir a Acum
0106 25 000F          AND AX,mascara ; Quita 4 bits de la palabra
0109 74 05            JZ SALIR   ; Salir si el resultado es cero
010B 8B 07            MOV AX,[BX] ; Guarda bits en dirección
```

```
                                 ; Apuntada por el registro BX
010D 43               INC BX     ; Incrementa contador datos
010E EB F4            JMP AQUI   ; Regresa por siguiente byte
0110      SALIR:                 ; Pasa a la siguiente instrucción
0110 codigo               ENDP
;-----------------------------------------------------------------
0110 ejemplo          ENDS
;*****************************************************************
*
    END inicio
Symbols-1
Segments and Groups:
N a m e   Length       Align    Combine Class
DATAREA . . . . . . . . . . . . 0502    PARA    NONE
EJEMPLO . . . . . . . . . . . . 0110    PARA    NONE
Symbols
N a m e   Type         Value     Attr
ADDR1 . . . . . . . . . . . . . L WORD  0500    DATAREA
AQUI . . . . . . . . . . . . . .        L NEAR  0104    EJEMPLO
CODIGO . . . . . . . . . . . . .        F PROC  0000    EJEMPLO
Length = 0110
INICIO . . . . . . . . . . . . .        L NEAR  0100    EJEMPLO
MASCARA . . . . . . . . . . . . NUMBER  000F
SALIR . . . . . . . . . . . . . L NEAR  0110    EJEMPLO
FILENAME. . . . . . . .     TEXT ensambla
64 Source Lines
64 Total Lines
10 Symbols
51078 + 264170 Bytes symbol space free
0 Warning Errors
0 Severe Errors
```

13.5 Direccionamiento a Memoria

Aunque realicemos solamente operaciones con los registros internos a la ***UPC***, eventualmente necesitaremos guardar la información en una localidad de memoria principal. Para realizar esta operación, se debe proporcionar, como operando, una dirección de la que se traerán o en la que se almacenarán los datos. Las formas de especificar la dirección dentro de la instrucción pueden ser muy variadas, pero contamos con tipos muy específicos como los que describimos a continuación.

13.5.1 Direccionamiento Implícito

Una instrucción que usa el direccionamiento implícito utiliza el contenido del registro contador de datos (***DC***) como dirección de memoria.

Esto implica un proceso de dos pasos:

1. Cargar la dirección de memoria requerida en el registro ***DC***.

2. Una instrucción sin operandos se ejecuta y ésta utiliza al registro ***DC*** como un apuntador a memoria.

13.5.2 Direccionamiento directo a memoria

Algunas instrucciones pueden especificar la dirección con la que trabajarán directamente en su operando; a éstas se les conoce como instrucciones con direccionamiento directo a memoria.

Toda computadora tiene instrucciones con un rango limitado de memoria en la que pueden actuar. Si todas las instrucciones de una computadora están limitadas por esta restricción, a la computadora se le conoce como ***paginada***.

Para formar la dirección final a la que la ***UPC*** direccionará, se usa la parte alta del Contador de Programas (***PC***) y se une a la dirección de página para formar la ***dirección efectiva***. Este término se aplica a toda dirección de memoria que se calcule de alguna forma, utilizando para ello la información provista por la misma instrucción.

Los bits tomados del ***PC*** son llamados ***número de página*** y los bits proporcionados por la instrucción forman la dirección dentro de la página. Combinando ambos tenemos la dirección efectiva.

La imposición más severa impuesta por un esquema de páginas fijas es que el programa no puede referirse a otra página y los programas no pueden residir en los límites de las páginas cruzándolas; así que el programador gasta mucho tiempo calculando esos límites y las localidades de memoria para su programa.

Un método para eliminar algunas de las restricciones del direccionamiento de páginas, consiste en proveer a la computadora de una página base y un registro de apuntador de página que pueda ser alterado por el programa a su conveniencia.

Una variación del esquema anterior consiste en realizar saltos relativos al ***PC*** sumando un desplazamiento contenido en la instrucción; a esta variación se le llama ***direccionamiento relativo***.

13.5.3 Direccionamiento Indirecto

Una última variación es la de usar una localidad de memoria que será utilizada como dirección para encontrar los datos que queremos llevar o recuperar de la memoria. Este esquema es flexible y permite usar apuntadores para aplicar todas las técnicas de

programación avanzada a las computadoras directamente (uso de apuntadores, colas, pilas, etc.).

13.6 Tipos de Instrucciones

A grandes rasgos las instrucciones pueden ser divididas en grupos muy genéricos; una vez más, depende del fabricante si incluye todos los tipos (o más) o un subconjunto de ellos. La aplicación final del producto dicta muchas veces el tipo de instrucción necesario y el superfluo.

Aunque no exhaustiva, la siguiente clasificación puede muy bien servir a nuestros propósitos:

1. Instrucciones que mueven datos. Incluyen instrucciones que mueven datos entre un registro y la memoria; entre dos registros e instrucciones inmediatas que cargan a los registros con un valor especificado en el operando de la instrucción.
2. Instrucciones aritméticas y lógicas. Son generalmente instrucciones que trabajan con dos operandos y realizan sumas, restas, multiplicaciones, divisiones, ***Y***, ***O,*** así como sumas y restas con acarreos. Las banderas generalmente se afectan con el resultado.
3. Corrimientos. Son de tres tipos: corrimientos, corrimientos aritméticos y corrimientos lógicos. En algunos casos la palabra se corre a través de la bandera de acarreo.
4. Saltos y subrutinas. Incluyen saltos incondicionales, saltos condicionales basados en las banderas, llamadas a subrutinas y regresos de subrutinas.
5. Incrementos y reducciones de los registros y la memoria. Usualmente suman o restan una unidad a la memoria o a los registros para llevar una cuenta en bucles o para acceder a localidades contiguas de memoria.
6. Comparaciones y pruebas. Comparan dos operandos y modifican el registro de banderas de acuerdo con el resultado o realizan pruebas sobre un operando.
7. Meter y sacar información de la pila (stack). Disminuyen o incrementan el registro de apuntador a la pila de acuerdo con la operación realizada: empuja o saca.
8. Instrucciones con las banderas de estatus. Limpian o fijan alguna de las banderas del registro de banderas de la ***UPC*** antes de realizar alguna operación; habilitan o deshabilitan las interrupciones a la ***UPC***.

9. Instrucciones especiales. Todas las que no caigan dentro de las 8 anteriores como son las de entrada y salida, que algunos prefieren clasificar en un grupo extra, y otras.

Puesto que las instrucciones son distintas entre las computadoras, se discute cada grupo en general y en el siguiente capítulo se da un ejemplo de una computadora y sus instrucciones.

Otra clasificación de las muchas existentes agrupa las instrucciones en 3 divisiones aún más genéricas:

1. Instrucciones de transferencia de datos que mueven información (direcciones, valores de operandos, instrucciones, etc.) sin cambio de una parte de la computadora a otra.
2. Instrucciones de procesamiento de datos que transfieren datos realizando las operaciones necesarias requeridas.
3. Instrucciones de control de programa usadas para determinar el orden de ejecución de las otras instrucciones.

13.7 Lenguajes de Alto Nivel

La programación a nivel máquina es necesaria para que ésta funcione, pero un usuario común y corriente no puede invertir tanto tiempo en aprender a programar una computadora cuando su objetivo principal es la resolución de problemas.

Los lenguajes de alto nivel son la parte del sistema operativo (aplicaciones) con la que el usuario avanzado se familiarizará más (se prefieren hoy en día las soluciones "enlatadas" en forma de programas de aplicación con cierta programación integrada: hojas de cálculo, procesadores de palabras, comunicaciones, autoedición, etc.). Para poder hacer que la computadora se comporte como queremos, debe haber una forma de comunicar nuestros deseos a la máquina. La mayoría de las computadoras existentes tienen alguna forma de lenguaje desarrollado para ellas. Existen tantos lenguajes como existen computadoras y cada uno de ellos llena un sitio en especial; aunque todos tienen en común que intentan ser de propósito general y "fáciles de usar".

Aunque es verdad que las diferencias de cada máquina nos limitan en su forma de programarla en lenguaje ensamblador (u objeto), esto no se aplica en su gran mayoría en los llamados lenguajes de ***alto nivel***. ¿Por qué el nombre de alto nivel? Pues porque comparados con los primeros métodos de programar una computadora

son mucho más potentes y flexibles. De los lenguajes de alto nivel más conocidos y entre los más viejos se encuentra el ***BASIC*** (Beginner's All-purpose Symbolic Instruction Code). Es el lenguaje de alto nivel más usado y popular en los sistemas pequeños de computadoras (computadoras personales) y en muchos de los sistemas grandes. Debe su popularidad a la sencillez de su sintaxis y a que es muy sencillo de aprender no tomando más que unas cuantas horas para poder realizar nuestro primer programa funcional.

Esta misma sencillez del lenguaje ***BASIC***, es su principal desventaja pues conforme las técnicas de programación han ido mejorando, se necesitan cada vez herramientas más poderosas que llenen los requerimientos; han surgido así una serie de lenguajes que se substituyen unos a otros y cada uno clama ser la última y mejor herramienta para la programación:

Lenguaje	Acrónimo	Uso
FORTRAN	**FOR**mula **TRAN**slator	Programación científica
COBOL	**CO**mmon **B**usiness **O**riented **L**anguage	Negocios
SMALLTALK		Objetos
C		Sistemas
SIMULA		Simulación
PASCAL		Enseñanza de técnicas correctas de programación
C++		Interfaces gráficas
y muchísimos otros más cada uno con sus méritos, ventajas y debilidades.		

13.8 Intérpretes y Compiladores

No importando qué tan sofisticado sea el programa o el lenguaje en que esté escrito, nuevamente llegamos al punto en que la ***UPC*** debe interpretarlo y ejecutarlo a la máxima velocidad posible. Esto implica traducir cada una de las instrucciones del lenguaje de alto nivel o ensamblador a unos y ceros que son colocados en las localidades precisas de memoria para que sean ejecutadas.

Las dos formas existentes de traducir las líneas de código de un lenguaje de alto nivel son usando programas intérpretes y programas compiladores.

13.8.1 Intérpretes

Un programa intérprete toma el código desarrollado por el usuario y escrito en un procesador de palabras o en un procesador de texto proporcionado, por el mismo intérprete. Al darle la orden de ejecutar un programa, traduce una a una las instrucciones del ***código fuente*** a lenguaje de máquina.

Esto tiene una serie de ventajas y desventajas:

Ventajas:

- Muy fácil corregir errores y desarrollar programas, pues en todo momento tenemos control sobre la ejecución; podemos interrumpir, poner puntos de espera, preguntar el contenido de variables, ejecutar el programa paso a paso, etc.
- Fácil de entender y manejar. Como no permite tantas opciones, no tenemos tantos comandos que aprender para controlar el proceso.
- Los errores se corrigen rápidamente y se ve de inmediato el resultado de estas correcciones.
- Las instrucciones del programa siempre están a disposición del usuario y éste puede modificarlas a su conveniencia.

Desventajas:

- Para la distribución del programa terminado debemos incluir el intérprete y las instrucciones para usarlo; además de las instrucciones para usar el programa.
- No da flexibilidad pues no permite unir el programa con otros desarrollados en otros lenguajes.
- No es portátil de un ambiente a otro pues el intérprete usualmente es distinto de máquina a máquina; a menos que se apegue estrictamente a un estándar.
- El proceso de traducir las instrucciones una a una lleva tiempo y si, por cualquier causa, es necesario ejecutar alguna instrucción que ya fue traducida, es necesario volver a traducirla una y otra vez.
- Se requiere de la memoria necesaria para poder ejecutar el intérprete, más la memoria necesaria para el programa.
- Como las instrucciones son interpretadas una a una, no es posible optimizar la traducción pues no tenemos el

concepto de programa completo, sino de instrucción en instrucción.

- Si se quiere vender el programa final, se debe incluir cada una de las instrucciones permitiendo que el programa sea copiado y/o modificado ilegalmente las veces que se desee.

Por todas las razones expuestas, es muchas veces preferible utilizar un compilador (ver siguiente sección) aunque algunas veces, si es posible, se prefiere desarrollar el programa en un intérprete y usar la versión final que entrega un compilador aprovechando las ventajas que dan estos auxiliares.

13.8.2 Compiladores

Un programa compilador ejecutará el mismo programa que un intérprete, pero la acción se realiza de un modo distinto en cada caso. El compilador toma un programa fuente como una unidad y lo traduce a programa objeto en varios pasos dejando un programa ejecutable que ya no requiere de ningún otro programa para ejecutarse, a excepción del sistema operativo.



Grace Murray Hooper (1906-1992)

Matemática estadunidense que desarrolló el primer compilador (A-0) y del lenguaje COBOL para la compañía Remington. Promovida a contralmirante en la II guerra mundial.

El programa que entrega el compilador puede ejecutarse directamente desde el procesador de comandos del sistema operativo. Para que esto suceda, deben existir varias condiciones:

- Que el programa fuente esté libre de errores de sintaxis
- Que el programa fuente no tenga errores de ejecución
- Que se le agreguen al programa fuente todas aquellas rutinas que necesita para poder realizar su trabajo.

Para que todo esto se cumpla se requiere de un trabajo adicional por parte del usuario y un esfuerzo mucho mayor que en el uso de un intérprete.

Es tal la popularidad de los sistemas de computación y de los lenguajes, que las compañías que los fabrican han invertido mucho tiempo y dinero para entregar un producto que sea rápido y evite muchas de las penurias que se encontraban en los primeros compiladores comerciales.

El ciclo de desarrollo de un programa con un compilador consiste en escribir el programa en un procesador de texto separado del programa y ejecutar el compilador. Si no existen errores de

sintaxis se procede a ligar con las librerías de funciones que el programa requiere durante su ejecución; se ejecuta el programa objeto y si existe algún error, por pequeño que este sea, se repite todo el proceso.

Como se ve de la descripción anterior el proceso es minucioso, elaborado, complicado y tardado. Debe existir alguna ventaja en usar un compilador para que éste exista. La principal ventaja que se obtiene es la rapidez de ejecución del programa final. Pero siguen existiendo ciertas ventajas y desventajas con respecto a un intérprete:

Ventajas:

- Programa objeto muy rápido en su ejecución (hasta 100 veces más rápido que con un intérprete dependiendo de las instrucciones y otros factores).
- Se puede unir el programa con otros desarrollados en otros lenguajes, por lo que tenemos acceso a una librería desarrollada por otros.
- Las instrucciones son traducidas una única vez.
- No existen pedazos de código con errores potenciales pues el programa se traduce como un todo y no instrucción a instrucción. Al traducir instrucción a instrucción puede ser que existan segmentos del programa que nunca se ejecuten en nuestras pruebas.
- Se puede optimizar el resultado final pues se conocen todas las referencias y todo lo que hace el programa en términos de instrucciones.
- No se requiere de ningún programa auxiliar para poder ejecutar el producto terminado.

Desventajas:

- El ciclo de desarrollo es tardado: programar, compilar, ligar y ejecutar. Mientras que en el intérprete sólo hay un paso: ejecutar.

Como la única desventaja aparente es la del desarrollo, los compiladores modernos ya incluyen un ambiente integral en el que se escribe el programa, se compila, liga y ejecuta en un sólo paso. Para ello se utiliza la memoria y sólo se graba el programa final como un módulo ejecutable cuando el ciclo de desarrollo está completo y así lo solicitamos. Estos productos tratan de combinar

las ventajas de un intérprete con las de un compilador y eliminar la mayoría de las desventajas de los intérpretes.

En la siguiente tabla (tabla *13.1*) están resumidas algunas de las ventajas y desventajas de cada uno de los esquemas para producir el código utilizable por una computadora.

Tabla 13.1

Ventajas y Desventajas de Lenguaje de Bajo Nivel, Interpretes y Compiladores

Lenguaje de Bajo Nivel	Interprete	Compilador
Requiere menor espacio de memoria para ejecutarse: se necesitan menos instrucciones de bajo nivel para realizar la misma tarea (no hay instrucciones redundantes producidas por un traductor externo).	Requiere de mucho espacio en memoria para poder ejecutarse: genera muchas instrucciones redundantes (dependiendo del interprete).	Requiere de espacio extra en memoria para poder ejecutarse: genera instrucciones redundantes (dependiendo del compilador).
Control preciso del hardware: algunas instrucciones de bajo nivel están diseñadas específicamente para controlar el hardware a bajo nivel.	El control del hardware no siempre es posible y se depende de rutinas de 3eros para poder lograrlo.	El control preciso del hardware es difícil, aunque se puede lograr combinando rutina en otros lenguajes o en ensamblador.
Se produce un programa objeto.	No se produce un programa objeto.	Se produce un programa objeto.
Es difícil encontrar los errores.	Es fácil encontrar los errores.	Es difícil encontrar los errores.
La computadora sólo requiere del programa objeto.	El intérprete debe estar presente para poder ejecutar el programa.	La computadora sólo requiere del programa objeto.
Se debe traducir todo el programa sin errores antes de poderse ejecutar.	Se puede ejecutar el programa, aunque tenga errores.	Se debe traducir todo el programa sin errores antes de poderse ejecutar.
Más eficiente: Las funcionalidades del hardware se aprovechan al máximo (como, por ejemplo, el ancho del bus).	El programa se ejecuta relativamente lento pues cada instrucción se traduce a lenguaje de máquina.	El programa se ejecuta relativamente rápido pues cada instrucción ha sido previamente traducida a lenguaje de máquina.
Un cambio en el programa requiere compilar todo.	Un cambio en el programa no requiere compilación previa.	Un cambio en el programa requiere compilar todo.

Las instrucciones son limitadas y su uso es complicado.	El conjunto de instrucciones es muy amplio y fácil de entender y usar aún para un neófito.	El conjunto de instrucciones es muy amplio y fácil de entender y usar.

13.9 Sistemas Operativos Avanzados

La evolución de las computadoras sigue aún un ritmo desenfrenado. Muchas cosas sorprendentes son realizadas con las computadoras y quedan aún muchas más por realizar: Computadoras gigantescas que realizan todas las transacciones de un banco incluyendo a sus cajeros automatizados; transacciones internacionales de computadora a computadora; computadoras para las aerolíneas que controlan los vuelos y reservaciones a nivel mundial; computadoras de tráfico aéreo que llevan el control vía radar de todos los vuelos que entran y salen de un aeropuerto y muchas aplicaciones más.

¿Puede usted imaginar la cantidad de potencia de cómputo requerido para esto, la complejidad de un sistema operativo y los programas necesarios para cumplir con el cometido asignado a la computadora?

Todo esto puede atribuirse al nacimiento de grandes e increíblemente complejos sistemas operativos, todos ellos imprácticos hace apenas unos cuantos años. Hasta hace poco tiempo el uso de más de 64K bytes de memoria en un sistema personal era impensable e incosteable. Hoy son contados los sistemas operativos que caben en esa cantidad de memoria.

La tendencia sigue evolucionando hacia compartir los datos por lo que términos como multitareas, multiusuario y multiproceso no serán del todo desconocidos en un futuro. La necesidad de velocidad es tanta que nuevas arquitecturas surgen como alternativas del uso de un sólo procesador central: arquitectura paralela y redes neuronales son la tendencia tecnológica del futuro.

13.10 Resumen

Una de las partes primordiales de un sistema de cómputo y sin el cual su funcionamiento no es posible, es el de la programación. Por medio de un programa podemos guiar todas las acciones de la electrónica para poder llegar a una aplicación práctica y útil.

El aspecto de la programación es muy extenso e incluye muchos temas complejos. Pero en su aspecto fundamental se describe brevemente en este capítulo.

El sistema operativo (***SO***) forma la parte sobre la cual actúan todos los demás programas; todo requerimiento a la computadora es encauzado por el canal adecuado por medio de la parte de entrada y salida básica del ***SO*** llamada ***BIOS***.

Para poder ser ejecutado todo programa debe residir en la memoria y colocarse en ella de alguna forma gracias a comandos que la ***UPC*** pueda entender e interpretar; para ello se han desarrollado toda una serie de programas llamados intérpretes y compiladores que se encargan de esta labor.

13.10.1 Puntos Importantes del Capítulo

- Un programa de arranque inicia el sistema de cómputo y le permite funcionar.
- El procesador de comandos de un sistema operativo queda residente todo el tiempo para recibir órdenes de otros programas o del mismo usuario.
- Un programa consiste en una secuencia lógica de instrucciones y programar significa crear estas secuencias.
- El código fuente consiste en las instrucciones que el programa ensamblador interpretará para crear un código objeto.
- El código objeto es el resultado de un ensamblador y consiste en el programa fuente interpretado y colocado en la memoria para su ejecución.
- Toda Unidad de Procesamiento Central (***UPC***) consta de un juego de instrucciones definidas por el fabricante durante su construcción.
- Entre las formas de indicar una localidad o dirección de memoria contamos con: direccionamiento directo, implícito o indirecto.
- Las instrucciones que permite una ***UPC*** son catalogadas de acuerdo con su función y existen varias de estas clasificaciones.
- Un intérprete traduce instrucción a instrucción un programa fuente mientras que un compilador lo hace todo de una vez.

13.11 Problemas

13.1 Investigue qué programa ensamblador viene junto con los circuitos basados en una ***UPC*** de Motorola, tales como los Apple.

13.2 Encuentre el conjunto completo de instrucciones de una ***UPC*** 68000 de Motorola.

13.3 Haga un programa sencillo de multiplicar dos números de 8 bits en lenguaje ensamblador inventado por usted.

13.4 Realice el mismo problema anterior pero ahora con lenguaje ensamblador del circuito ***UPC*** 80486 de Intel.

13.5 Ensamble a mano el programa del problema *13.4*.

13.6 Usando un ensamblador comercial, ensamble el programa del problema *13.4* y compare con el del *13.5*.

13.7 Compile el siguiente programa en C:

```
int main()
{
    return 0;
}
```

que, obviamente no hace nada. Use, para ello un compilador minimalista tal como *Tiny C*[48]. A continuación, desensamble el código resultante usando una herramienta, por ejemplo, *GDB*[49]. Si cuenta con Visual Studio® de Microsoft® puede hacer ambas operaciones en dicho compilador. ¿Podría optimizar el código de alguna forma? Recuerde que para poner un registro en cero de forma rápida y eficiente basta con usar:

```
xor eax,eax
```

Como alternativa para desensamblar el programa puede, también, usar DEBUG[50].

---

[48] Desarrollado hasta 2018 por Fabrice Bellard de forma gratuita.
[49] Versión actualizada disponible en 2020 en versiones de 32 y 64 bits de forma gratuita.
[50] Disponible gratuitamente en Windows® de Microsoft® en todas sus versiones (siendo Windows 2010 la más actual al momento de la impresión de este libro) pero sólo en sus gamas de 32 bits.

14 Una Unidad de Procesamiento Central Comercial

14.1 Un ejemplo: El circuito 80486 de Intel

Hemos escogido para presentar como ejemplo de la ***UPC*** al circuito 80486 de Intel[51]. Esta selección obedece al hecho de que, en México como alrededor del mundo, casi todas las computadoras personales se basan de una forma u otra en este circuito o en uno del mismo fabricante y familia (todas las computadoras compatibles con IBM y más recientemente las Apple que cambiaron su diseño tradicional basado en el ***CPU***, diseñado por la compañía Motorola, a los de la competencia diseñados por Intel).

Las microcomputadoras han encontrado su camino hacia muchas de las aplicaciones reservadas con anterioridad a las minicomputadoras y supercomputadoras. El abaratamiento de los circuitos y su incesante popularidad ha dado nacimiento a circuitos muy avanzados tecnológicamente. El circuito 80486 (y los de su familia) cuentan entre sus características con:

- Arquitectura interna de 32 bits.
- Uso de estructura de instrucciones *CISC* (ver el capítulo *11* para más precisiones).
- Soporte de más de 16MB de memoria principal (bus de dirección de 30 líneas = 2^{30}= 4 GB).
- Aritmética de 8 y 16 bits con y sin signo, binario y decimal; incluye multiplicación y división.
- 14 palabras de registros de 16 bits.
- Capacidad de acceso directo a memoria.
- Soporte de coprocesadores (para el modelo SX, el modelo DX lo incluye en el propio ***CPU***).
- Operaciones con cadenas.
- Soporte de E/S por medio de mapa de memoria.
- Unidad de punto flotante (*FPU*) incluida en el mismo chip del ***CPU*** con su propio bus.

Los registros generales de la ***UPC*** han aumentado conforme los requerimientos de cómputo han evolucionado. En un circuito de

[51] Circuito en desuso, pero ejemplo práctico que se puede usar como base para comprender otros diseños más modernos.

este tipo encontramos 14 registros de 32 bits (de los cuales sólo mencionaremos los 16 bits bajos):

Tabla 14.1	
Registros	
Registros generales	
AX	Acumulador
BX	Base
CX	Cuenta
DX	Datos
Grupo de apuntadores e índices	
SP	Apuntador de pila (stack)
BP	Apuntador base
SI	Apuntador fuente
DI	Apuntador datos
IP	Apuntador de instrucción
F	Banderas del sistema
Registros segmentos	
CS	Segmento de código
DS	Segmento de datos
ES	Segmento extra
SS	Segmento de pilas

El grupo de registros generales consta de 32 bits al igual que los demás, pero puede leerse en dos partes de 16 bits. A su vez la parte baja de 16 bits se puede leer en dos partes: una baja especificada con el sufijo *L* y una alta especificada con el sufijo *H*. Por ejemplo, el registro *AX* se puede dividir en *AL* y *AH*.

Algunos de estos registros tienen usos específicos y algunas instrucciones requieren de ciertos registros:

Tabla 14.2	
Registros y su uso	
Registro	**Operación**
AX	Multiplica o divide palabras
AL	Multiplica o divide bytes
AH	Multiplica o divide bytes
BX	Traduce
CX	Operaciones con cadenas o contador en bucles
CL	Desplazamiento y rotación

DX	Multiplica o divide palabras
SP	Operaciones con la pila
SI	Operaciones con cadenas
DI	Operaciones con cadenas
IP	No es accesible más que con saltos

El registro de banderas consta de 9 bits como se muestra en la tabla *14.3*. Las decisiones de la computadora se basan muchas veces en el estado de estos bits.

Tabla 14.3	
Banderas	
Bandera	**Función**
OF	Saturación
DF	Dirección (operación con cadenas)
IF	Habilitación de interrupciones
TF	Trampa (ejecución paso a paso)
SF	Signo (resultado negativo)
ZF	Cero
AF	Acarreo auxiliar
PF	Paridad (par)
CF	Acarreo

Este circuito ve a la memoria como un grupo de unidades lógicas llamadas segmentos. Cada segmento puede ser de hasta 64K bytes de longitud y son controlados por los registros de segmentos; cada uno indicando el final del segmento en particular (ver figura *14.1*).

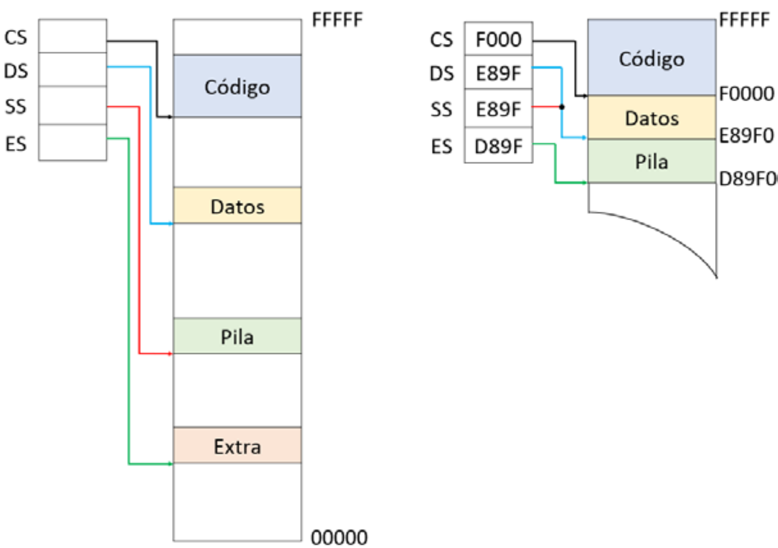


Figura 14.134 Los cuatro registros de segmento y su uso.

Sólo existe una restricción con respecto a los segmentos: deben comenzar en un párrafo que es una dirección que es divisible exactamente entre 16. Para ello, los 4 bits menos significativos deben ser cero.

Los segmentos pueden ser adyacentes, disjuntos, sin traslape o con traslape parcial o total. Si requerimos usar un registro como apuntador a memoria, necesitaremos sumar un desplazamiento para poder completar las 32 líneas de direccionamiento disponibles. Este desplazamiento se indica en la tabla *14.4*.

Tabla 14.4			
Uso del Desplazamiento			
Tipo de referencia	**Segmento base por omisión**	**Segmento base alterno**	**Desplazamiento**
Traer instrucción	CS	Ninguno	IP
Operación con pila	SS	Ninguno	SP
Variables (excepto los que siguen)	DS	CS, ES, SS	Dirección efectiva
Fuente cadena	DS	CS, ES, SS	SI
Destino cadena	ES	Ninguno	DI
BP usado	SS	CS, ES, SS	Dirección

como registro base			efectiva

El circuito cuenta con una cola de instrucciones donde el procesador va guardando las siguientes instrucciones a ejecutar disminuyendo el ciclo de instrucción del típico traer instrucción, decodificar instrucción y ejecutarla a sólo dos pasos; mientras se interpreta la instrucción se trae la siguiente al mismo tiempo.

14.2 Formas de Direccionamiento

Los lenguajes de alto nivel utilizan variables en lugar de los registros para realizar sus operaciones. Las variables pueden ser simples o complejas tales como matrices, registros, etc. El circuito tiene varias formas de direccionamiento que permiten acceder a estos tipos de variables proveyendo un desplazamiento a la localización física de la variable en memoria.

Como observamos en la tabla *14.5*, existen varios métodos para acceder a los datos de las variables

Tabla 14.5	
Formas de Direccionamiento	
Forma	**Localización de los datos**
Inmediato	En la misma instrucción
Registro	En el registro
Directo	En la localidad de memoria apuntada por el desplazamiento contenido en la instrucción
Indirecto de Registro	En la localidad de memoria apuntada por el desplazamiento contenido en el registro
Indexado o base	En la localidad de memoria apuntada por la suma del índice o registro base y el desplazamiento contenido en la instrucción
De base e índice con desplazamiento	La dirección de memoria se forma por la suma del contenido del registro base; el contenido del registro índice y un desplazamiento

Como registros bases podemos utilizar *BP* o *BX*; como registros subíndices *SI* o *DI*. Los tipos de direccionamiento de la tabla a partir del directo pueden soportar tipos de datos en forma de arreglo, registro o variable encontrados en los lenguajes de alto nivel (ver figura *14.2*).

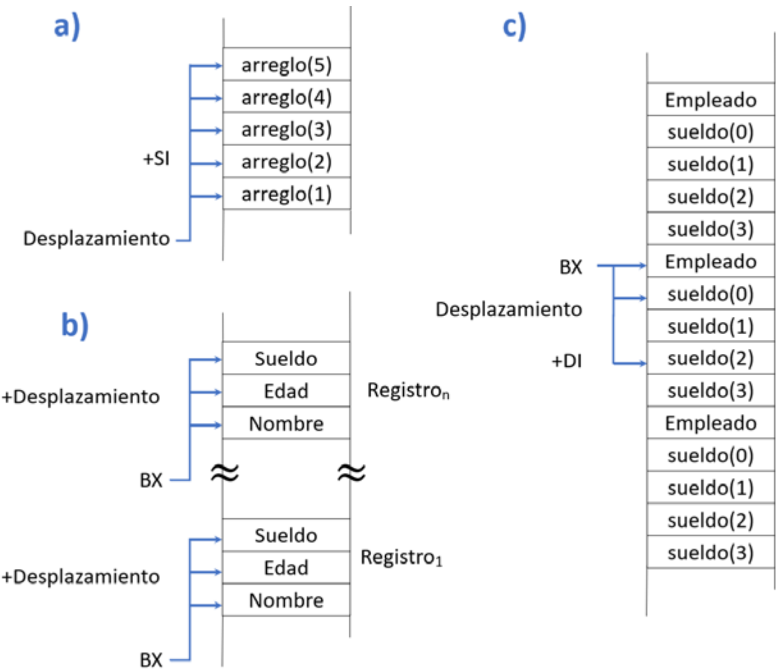


Figura 14.135 Formas de direccionamiento.

El método de base e índice con desplazamiento puede acceder a arreglos dentro de registros simulando los registros de una base de lenguajes de alto nivel (Pascal, C, Dbase, PHP, .NET, etc.) como se ve en la figura *14.2c*.

14.3 Ventajas de una Memoria Segmentada

Acceder a la memoria como un segmento dentro del mapa completo de memoria nos ofrece varias ventajas sobre el acceso de forma lineal donde se ve a la memoria como una matriz continua de localidades.

El acceso lineal exige la construcción de una sola dirección en oposición a la construcción por partes de una memoria segmentada. Si tenemos una longitud de dirección de 32 bits, tendríamos que contar con registros de por lo menos esa cantidad para poder guardar una dirección o utilizar varios de ellos. Si usamos una memoria segmentada, la dirección es construida a partir de varios registros lo que nos da más flexibilidad y permite un desarrollo a futuro, no exigiendo ampliar la longitud de los registros si no escogemos hacerlo.

Los programas pueden armarse y localizarse en memoria en un área libre con sólo modificar los apuntadores *CS* y *DS* para apuntar a los sitios escogidos. Esta modalidad permite también un diseño

modular de los programas no alcanzada con los diseños anteriores de mapas de memoria.

Una memoria segmentada facilita la administración de la memoria y su protección, sumamente importante en los sistemas operativos modernos que realizan tantas tareas simultáneas (en apariencia). Diseños futuros pueden ir un paso más allá al incluir una tabla de los derechos de los programas a acceder distintas áreas de memoria sin cambiar el diseño de la arquitectura.

Presentamos un resumen de las ventajas de la memoria segmentada en la tabla *14.6*

Tabla 14.6
Ventajas de la Memoria Segmentada

Lineal	**Segmentada**
Se debe proporcionar toda la dirección	Se proporcionan las direcciones en dos o más partes
Se limita el desarrollo futuro de memorias más amplias	
No hay relocalización dinámica de programas	Sí hay relocalización dinámica de programas; lo que permite multitareas
No hay programación modular	
No hay manejo de memoria por lo que la protección de ésta no es fácil	

En los circuitos de la familia 80x86 la dirección física se construye sumando un desplazamiento a cualquiera de los registros ***CD***, *SS*, *DS* o *ES* con sus primeros 4 bits colocados en cero.

14.4 Instrucciones

Dentro de las computadoras, las instrucciones se dividen en varios tipos de acuerdo con su función. Ya en el capítulo anterior analizamos los grupos de instrucciones de acuerdo con una clasificación. En nuestra Unidad de Procesamiento Central de ejemplo dividimos las instrucciones en los siguientes grupos genéricos:

1. Movimientos de registros y datos
2. Multiplicación y división entre números decimales
3. Suma, resta
4. Corrimientos

5. Operaciones con cadenas
6. Traducción de bytes por tablas
7. Interrupciones programables
8. Bucles de programas
9. Coordinación de multiproceso y coproceso
10. Prueba no destructiva de bits

Las instrucciones pueden ser sin operandos (modo inmediato), con uno o dos operandos. Por ejemplo, la instrucción ADD BX, AX significa: suma el destino y la fuente (*BX* y *AX*) y deja el resultado en el destino (*BX*).

La mayoría de las instrucciones operan ya sea con bytes o palabras (dos bytes). En la tabla 14.7 especificamos las formas de direccionamiento utilizadas con las instrucciones y dónde se localizan los datos.

Tabla 14.7

Formas de Direccionamiento y Localización de los Datos

Modo	**Localización**	**Ejemplo**
Inmediato	En la instrucción	ADD CH, 5F
Registro	En el registro	ADD BX, DX
Directo	En la localidad de memoria apuntada por el desplazamiento contenido en la instrucción	ADD variable, BX
Registro	En la localidad de memoria apuntada por el desplazamiento contenido en el registro	ADD CX, [BX]
Índice o base	En la localidad de memoria apuntada por la suma del contenido del registro índice o base y el desplazamiento contenido en la instrucción	ADD [SI+6], AL
Base e índice con desplazamiento	La dirección de memoria es la suma del contenido del registro base, más el contenido del registro índice y un desplazamiento	ADD [BX+DI+5], DX

En la mayoría de las operaciones, se puede designar a los registros o contenido de la memoria como fuente o destino, sin embargo, los datos inmediatos sólo pueden ser fuente; de esta forma, se puede sumar 5 a una localidad de memoria y almacenar el resultado de nuevo en la memoria. Esto también ayuda a manipular variables en lenguajes de alto nivel, pues en lugar de escribir un pequeño programa con varios pasos para traer la variable de memoria a un registro para realizar la suma y luego regresarla a su lugar, la operación es realizada con una sola instrucción.

14.5 Instrucciones de Transferencia

Las instrucciones de transferencia, que se listan en la tabla *14.8*, nos permiten preparar los registros internos a la ***UPC*** o las localidades de memoria; comunicarnos con los puertos de Entrada/Salida, empujar y sacar datos de la pila y traducir datos de una forma a otra. Sólo dos instrucciones: ***POPF*** y ***SAHF*** afectan el registro de banderas.

Tabla 14.8

Instrucciones de Transferencia

Mnemónico	**Descripción de su operación**
MOV destino, fuente	Mueve byte o palabra
PUSH fuente	Empuja palabra a la pila
POP destino	Saca palabra de la pila
PUSHF	Empuja banderas a pila
POPF	Saca banderas de la pila
XCHG op1, op2	Intercambia el operador 1 con el operador 2
LAHF	Carga registro AH con las banderas
SAHF	Guarda en las banderas lo que hay en el registro AH
IN acc, puerto	Carga en el acumulador con un byte o palabra del puerto
OUT puerto, acc	Traspasa del contenido del acumulador hacia el puerto
LEA destino, fuente	Carga dirección efectiva
LDS destino, fuente	Carga apuntador en destino y registro DS
LES destino, fuente	Carga apuntador en destino y registro ES
XLAT	Traduce byte

La instrucción ***MOV*** (mueve) es una copia no destructiva del operando fuente (el segundo) al destino. La fuente y el destino son bytes o palabras y puede ser registros, localidades de memoria o variables, usando los métodos de direccionamiento disponibles. Fuente y destino no pueden ser a la vez contenidos de memoria. Por ejemplo, podemos usar la instrucción:

```
MOV calif, 6
```

Para mover un seis a *calif* que es un desplazamiento en memoria donde guardamos una variable.

Las instrucciones ***PUSH*** (empuja a pila) y ***POP*** (saca de pila) proveen una forma de guardar información en la pila de la que se lleva el control por medio de los registros apuntadores ***SS*** que apunta al fondo de la pila u origen, y ***SP*** que punta a la parte alta de la pila (ver figura *14.3*). Todas las operaciones con la pila son de 16 bits por lo que ***SP*** se incrementa o disminuye de dos en dos. En la figura *14.3* vemos que una instrucción ***PUSH*** disminuye ***SP*** en 2 y luego escribe la palabra en memoria, mientras que la operación ***POP*** incrementa ***SP*** y lee la palabra de memoria. Las palabras se escriben con la convención de byte bajo primero, byte alto después.

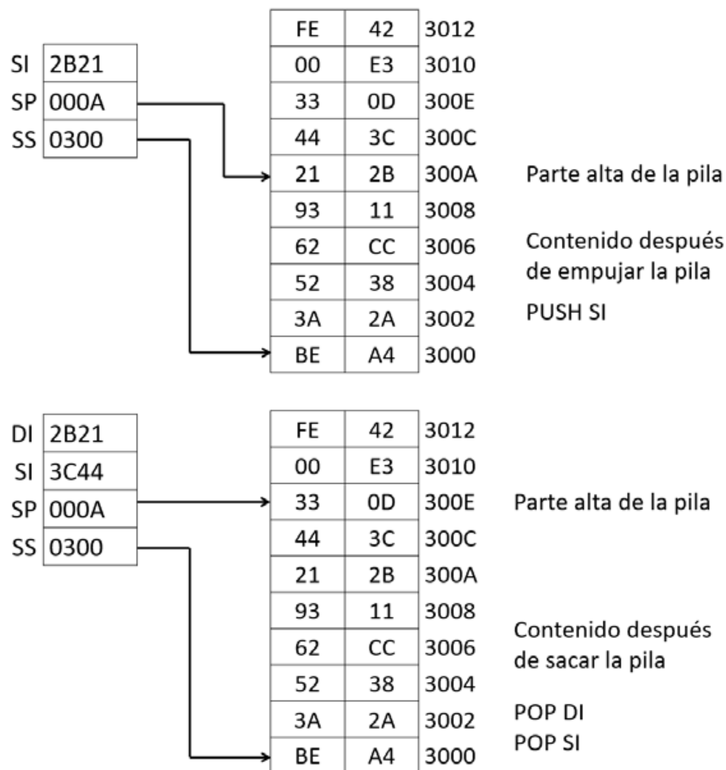


Figura 14.136 Instrucciones PUSH y POP.

Mostramos un ejemplo de la operación ***MOV*** para inicializar los registros; nótese que ***SP*** se inicializa inmediatamente después que ***SS*** pues la ***UPC*** deshabilita las interrupciones por una instrucción si se mueven datos de o hacia ***SS***. Refiérase a la figura *14.4* para el mapa de memoria resultante.

```
MOV AX,2000 ;Usa los 16 bits más significativos
MOV DS,AX   ;del área de datos para mover a DS
MOV AX,6534 ;Hacer lo mismo para la pila
MOV SS,AX   ;Sigue un movimiento de SS
MOV SP,0200 ;con uno de SP (interrupción habilitada)
JMP inicio  ;Carga CS e IP con la dirección de la
            ;1era instrucción del programa a ejecutar
```

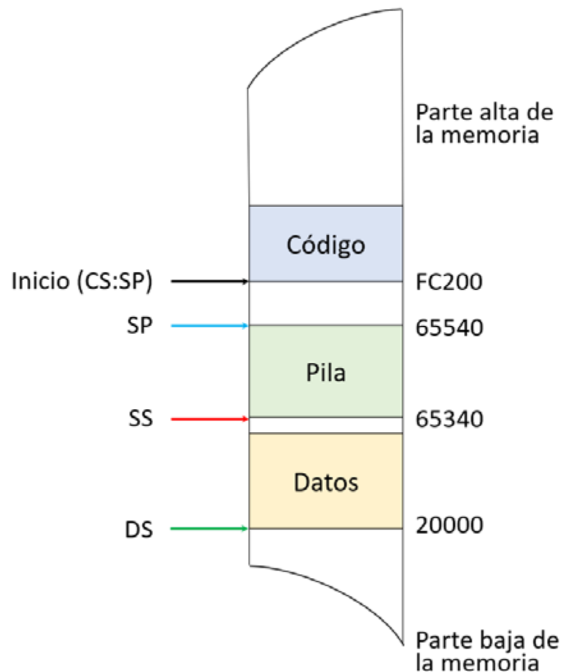


Figura 14.137 Inicialización de los segmentos.

Las instrucciones ***IN*** (acepta información de un puerto) y ***OUT*** (saca información a través de un puerto) se utilizan para la comunicación entre puertos de ***E/S*** en un espacio distinto al reservado de la memoria. Con estas instrucciones podemos direccionar 65,536 (2^{16}) puertos distintos. Se puede usar la parte baja del acumulador o todo el acumulador. Si se utiliza todo el registro del acumulador, el puerto debe ocupar 2 localidades consecutivas para poder realizar la transferencia en dos bytes. El acceso al puerto se hace de forma directa utilizando un byte (255 puertos distintos) o de forma indirecta por medio del registro ***DX*** con el que podemos tener acceso a los 65,536 puertos completos.

Las operaciones con cadenas y los procedimientos utilizan las instrucciones ***LDS*** (carga apuntador en ***DS***) y ***LES*** (carga apuntador en ***ES***) para inicializar sus apuntadores a un área de trabajo en la memoria.

Usualmente se requieren traducir datos en la computadora de una forma a la otra: ***BCD*** a ***ASCII***, Fahrenheit a Centígrados, etc. La instrucción ***XLAT*** (traduce) simplifica enormemente esta tarea. En la figura *14.5* damos un ejemplo de la traducción con el siguiente programa:

```
;Acepta un dato del puerto de entrada (seguramente el teclado)
;localizado en 0100, y convierte los datos de 0 a F hexadecimal
```

```
;a su equivalente ASCII guardados en una tabla
          LEA BX,tabla     ;Desplazamiento de la tabla en BX
          MOV DX,0100      ;DX contiene el número de puerto
          IN AL,DX         ;Toma los datos del teclado
                           ;usando entrada de forma indirecta
          XLAT             ;Usando los datos en AL como
                           ;un desplazamiento toma el valor ASCII
                           ;y coloca de nuevo en AL
```

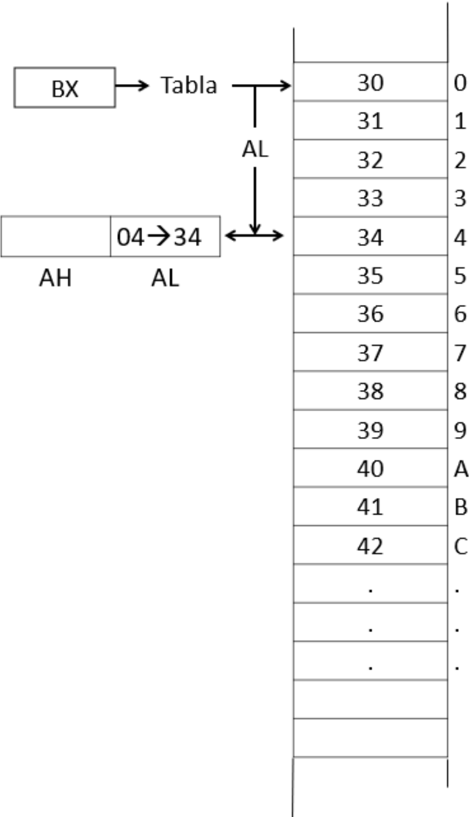


Figura 14.138 Uso de la instrucción XLAT.

14.6 Instrucciones Aritméticas

En la tabla *14.9* listamos las instrucciones aritméticas y su descripción. De especial atención aquí es la instrucción ***CMP*** (compara) que realiza la misma función que ***SUB*** (resta) pero el resultado no se coloca en ningún lado; sólo sirve para modificar las banderas y poder realizar un salto condicional después de la operación.

Las instrucciones ***MUL*** (multiplica) y ***DIV*** (divide) nos dan una herramienta poderosa para las operaciones que antes debían ser realizadas por medio de un programa completo. Estas operaciones se realizan en registros dedicados, como se muestra en la figura *14.6*.

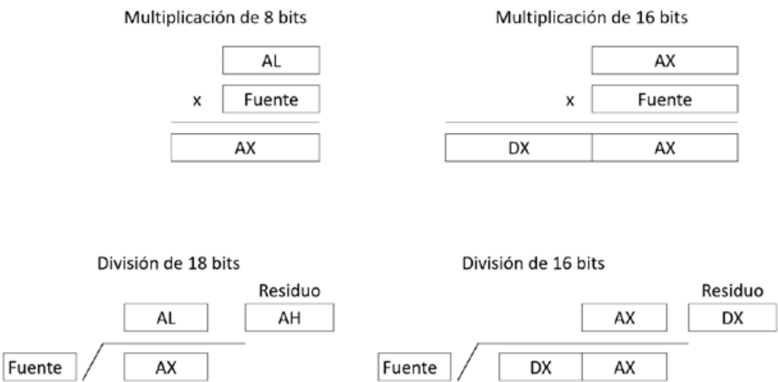


Figura 14.139 La división y multiplicación.

Los resultados de las operaciones se pueden ajustar con las instrucciones ***CBW*** (convierte byte a palabra) y ***CWD*** (convierte palabra a doble palabra). Mostramos un programa para demostrar cómo se debe realizar la operación completa:

```
; Este programa realiza la siguiente operación:
; (BH x BL) + CH/CL donde:
; BH = FB(-05), BL = FE(-02)
; CH = EC(-20), CL = 05(+05)
          MOV AL,BH    ;AX=?? FB nótese que sólo AL es cargado
          IMUL BL      ;AX= 000A(+10)= -5 x -2= BH x BL
          MOV DX, AX   ;salvar AX en DX provisionalmente
          MOV AL, CH   ;AX= 00EE(-20)
          CBW          ;AX= FFEC (convertido a palabra)
          IDIV CL      ;AX= 00FC(-4) en AL
          CBW          ;AX= FFFC (convertido a palabra)
          ADD AX,DX    ;AX= 0006(+6)
```

Si el resultado de la división no cabe en el byte ***AL*** o en la palabra ***AX***, una interrupción del tipo 0 (ver secciones *14.8* y *14.10*) es generada, donde se puede atender esta eventualidad.

Tabla 14.9	
Instrucciones Aritméticas	
Mnemónico	**Descripción de la Operación**
ADD destino, fuente	Suma byte o palabra
ADC destino, fuente	Suma byte o palabra con acarreo
SUB destino, fuente	Resta byte o palabra
SBB destino, fuente	Resta byte o palabra con préstamo
INC destino	Suma 1 a byte o palabra
DEC destino	Resta 1 de byte o palabra
NEG destino	Niega byte o palabra (complemento a 2)
CMP destino, fuente	Compara byte o palabra
MUL fuente	Multiplica byte o palabra sin signo
IMUL fuente	Multiplica enteros (byte o palabra)

DIV fuente	Divide byte o palabra sin signo
IDIV fuente	Divide enteros (byte o palabra)
CBW	cambia byte a palabra
CWD	Cambia palabra a doble palabra
DAA	Ajuste decimal para la suma
DAS	Ajuste decimal para la resta
AAA	Ajuste de ***ASCII*** para la suma
AAS	Ajuste de ***ASCII*** para la resta
AAM	Ajuste de ***ASCII*** para la multiplicación
AAD	Ajuste de ***ASCII*** para la división

14.7 Instrucciones de Manipulación de Bits

Estas instrucciones realizan las operaciones lógicas, así como las rotaciones y corrimientos de bytes o palabras en los registros o en la memoria. En la tabla *14.10* listamos las operaciones.

Tabla 14.10	
Operaciones Lógicas	
Mnemónico	**Descripción de la Operación**
NOT destino	Niega byte o palabra (complemento a 1)
AND destino, fuente	Operación ***Y*** con byte o palabra
OR destino, fuente	Operación ***O*** con byte o palabra
XOR destino, fuente	Operación ***O EXCLUSIVA*** con byte o palabra
TEST destino, fuente	Prueba byte o palabra
SHR destino, cuenta	Corrimiento lógico a la derecha de byte o palabra
SAR destino, cuenta	Corrimiento aritmético a la derecha de byte o palabra
SHL/SAL destino, cuenta	Corrimiento a la izquierda de byte o palabra
ROR destino, cuenta	Rotación a la derecha de byte o palabra
RCR destino, cuenta	Rotación a través del acarreo de byte o palabra
ROL destino, cuenta	Rotación a la izquierda de byte o palabra
RCL destino, cuenta	Rotación a la izquierda a través del acarreo de byte o palabra

De la tabla anterior destaca la operación ***TEST*** (prueba) que es una operación no destructiva idéntica a la ***AND*** (operación ***Y***) que afecta exclusivamente a las banderas. Por ejemplo, para probar el byte apuntado por ***BX*** en su bit más significativo:

```
TEST [BX],8000
```

Si el bit más significativo es 1, la bandera ***ZF*** (cero) se limpia y luego se puede usar un salto condicional dependiendo de la bandera cero.

Las rotaciones y corrimientos son mostrados en la figura *14.7*. Se debe recalcar que ***SAR*** (corrimiento aritmético a la derecha) puede no ser el mismo resultado que la división entre dos, pues el resultado es redondeado. En el mismo caso se encuentran ***SAL*** (corrimiento aritmético a la izquierda) y ***SHL*** (corrimiento a la izquierda) con respecto a la multiplicación).

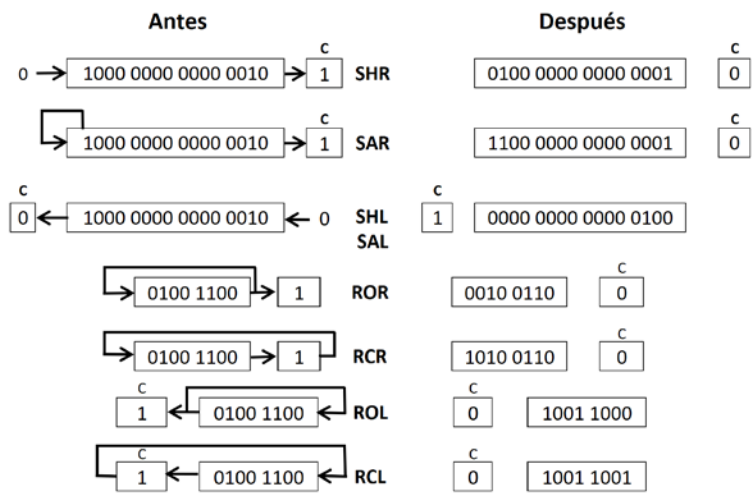


Figura 14.140 Instrucciones de rotación y corrimiento.

Todas estas operaciones de corrimientos y rotaciones toman la forma de:

Mnemónico registro/localidad de memoria, cuenta

donde la cuenta puede ser 1 o un número (hasta 255) localizado en la parte baja del registro ***CX*** (***CL***).

14.8 Instrucciones de Transferencias

Las instrucciones son traídas de la memoria utilizando el registro ***CS*** como el segmento e ***IP*** como el desplazamiento y la única

forma de modificar estos registros es indirectamente con las instrucciones de la tabla *14.11*.

Tabla 14.11	
Instrucciones de Transferencia	
Mnemónico	**Descripción de la Operación**
JMP destino	Salta a una localidad de memoria indicada en destino
Jcond destino	Salto condicional. Vea las condiciones en la tabla *14.12*
LOOP destino	Bucle
LOOPE/LOOPZ destino	Bucle mientras Sea igual/Sea cero
LOOPNE/LOOPNZ destino	Bucle mientras No sea igual/No sea cero
JCXZ destino	Salta si *CX=0*
CALL destino	Llama a una subrutina
RET (opcionalmente un valor)	Regresa de una subrutina
INT tipo	Interrumpe
INTO	Interrumpe en caso de saturación
IRET	Regresa de una interrupción

Tabla 14.12	
Saltos Condicionales	
Con signo	
JG/JNLE destino	Mayor/no menor no igual
JGE/JNL destino	Mayor o igual/No menor
JL/JNGE destino	Menor/No mayor no igual
JLE/JNG destino	Menor o igual/No mayor
JO destino	Saturación
JS destino	Signo
JNO destino	No saturación
JNS destino	No signo
Sin signo	
JA/JNBE destino	Arriba/No abajo o igual
JAE/JNB destino	Arriba o igual/No abajo
JB/JNAE destino	Abajo/No arriba o igual
JBE/JNA destino	Abajo o igual/No arriba
No importa el signo	
JC destino	Acarreo
JE/JZ destino	Igual/Cero

JP/JPE destino	Paridad/Paridad par
JNC destino	No acarreo
JNE/JNZ destino	No igual/No cero
JNP/JPO destino	No paridad/Paridad non

Los saltos pueden ser de dos tipos:

- Condicionales
- Incondicionales

Y a su vez se dividen en:

- Cortos. Saltos en el rango de 127 a −128 bytes. Se logran sumando un desplazamiento de 1 byte a *IP*.
- Cercanos. A no más de 32K bytes de distancia en forma circular. Esta forma contiene un desplazamiento de 2 bytes sumados a ***IP;*** puede ser usado también para moverse a todo lo largo del segmento.
- Lejanos (largos). Reemplazando ***CS*** e ***IP*** con los 5 bytes encontrados en la instrucción. Esto permite ir a un nuevo segmento.

Los saltos pueden ser tanto directos como indirectos. La memoria no se ve como una zona lineal sino como un área circular, esto permite que en los saltos cortos y cercanos no se salga uno del segmento. Si se encuentra uno casi al final de un segmento y se realiza un salto hacia adelante que intente salirse del segmento, la saturación del registro hace que esto no sea posible y en su lugar se realiza un salto hacia atrás: por ejemplo, FFFE+0005=0003.

Los saltos condicionales pueden ser divididos en tres grupos:

- Con signo. Usan la bandera de saturación y signo.
- Sin signo. Usan la bandera de acarreo.
- No importa el signo. Usan una bandera específica.

Mostramos un ejemplo para usar saltos por medio de una tabla:

```
tabla     DW error    ;Poner tabla de desplazamiento a rutinas
          DW rutina1  ;DW significa definir palabra
          DW rutina2
          DW rutina3
;La rutina acepta una entrada entre 1 y 3 ASCII del teclado
;localizado en el puerto 0 y salta a la rutina indicada por el
;número. Si el número no está en el rango de 1 a 3 se salta a una
;rutina de error.
inicio:   IN AL,0     ;acepta tecla del puerto 0
```

```
          SUB AL,30   ;-30 hexadecimal para cambiar ASCII
                      ;a número
          CMP AL,3    ;compara con 1,2,3
          JBE enrango ;si rango 1 a 3 estamos
                      ;apuntando correctamente
          XOR AL,AL   ;si no, poner AL en 0 para rutina de error
enrango:  XOR AH,AH   ;limpia parte alta de AX
                      ;para hacer una palabra
          MOV SI,AX   ;poner en registro índice para la dirección
          SAL SI,1    ;dobla valor para índice de
                      ; palabra y no byte
          JMP tabla[SI]    ;salta a subrutina apuntada por
                      ;tabla + desplazamiento de SI
error:                ;aquí irían las rutinas
rutina1:              ;la rutina del número 1
rutina2:              ;la rutina del número 2
rutina3:              ;la rutina del número 3
    .
    .
    .
```

Es muy importante poder realizar bucles donde se ejecute un grupo de instrucciones un número determinado de veces. Casi todas las ***UPC*** proporcionan una instrucción para poder lograr esto. Una forma de realizarlo sería restando la unidad de un contador y haciendo un salto condicional:

```
          DCR CX
          JNZ inicio
```

Se escogió, sin embargo, una sola instrucción: ***LOOP*** (bucle) que reemplaza a las dos anteriores y usa al registro ***CX*** como su contador. La instrucción sólo realiza saltos cortos.

Puesto que la instrucción ***LOOP*** primero disminuye el registro ***CX*** y luego realiza la comparación, podemos tener problemas si entramos a un bucle con 0 en el registro ***CX***, por lo que primero se debe asegurar que el programa verifique esta condición con la instrucción ***JCXZ*** (salta si registro ***CX*** es cero).

Las instrucciones ***LOOPE*** (bucle mientras igual) y ***LOOPNE*** (bucle mientras cero) verifican además las banderas de cero, saliéndose del bucle si ***CX*** llega a cero o si la bandera de cero es modificada. Para modificar la bandera puede usarse la comparación ***CMP*** antes de llegar al final del bucle.

14.9 Procedimientos o Subrutinas

Si se desea tener una programación eficaz y modular (aunque sea parcialmente) es necesario el uso de los procedimientos o subrutinas donde se ejecute un programa (procedimiento) en otra parte

de la memoria y una vez terminado éste, se regrese al punto de donde se le llamó desde el programa principal.

Las instrucciones utilizadas para realizar esta operación son ***CALL*** (llama) y ***RET*** (regresa). La instrucción ***CALL*** es muy similar a un salto, pero se salva la dirección de regreso (la dirección de la siguiente instrucción después de la de ***CALL***) en la pila. Esta instrucción puede ser tanto indirecta como directa, así como cercana o lejana. La llamada cercana sólo empuja ***IP*** a la pila mientras que la lejana empuja ***CS*** e ***IP*** a la pila, por lo que debe haber un regreso de procedimiento cercano o lejano que ajuste correctamente los valores a sacar de la pila.

Usualmente se requiere pasar parámetros a las subrutinas; esto se logra utilizando los registros, la pila o un área de memoria determinada.

Los procedimientos reentrantes son los que se llaman a sí mismos (o puede suceder que sea una rutina de servicio a una interrupción y que suceda la misma interrupción) y son de gran utilidad en ciertos tipos de algoritmos.

Damos a continuación un ejemplo de un programa que empuja los parámetros a usar en la pila:

```
;Programa que llama al procedimiento de suma
          LEA AX,arreglo   ;desplazamiento del inicio del arreglo
          PUSH AX
          MOV AX,10h       ;longitud del arreglo es de
                           ;16 palabras (10 hexadecimal)
          PUSH AX
          MOV AX,5h        ;valor a sumar al arreglo (5)
          PUSH AX
          CALL muestra;llama a la rutina
          .
          .
          .
;Procedimiento que suma un valor a un arreglo
;los parámetros son pasados en la pila de donde
;se toman
muestra:  PUSH BP          ;salva BP para protección
          MOV BP,SP        ;BP apunta a la cima de la pila
          MOV AX,[BP+4]    ;En AX número a sumar
          MOV CX,[BP+6]    ;En CX longitud del arreglo
          MOV SI,[bp+8]    ;En SI dirección del arreglo
          JCXZ alto        ;Si longitud del arreglo=0 regresa
otro:     ADD [SI],AX      ;Suma número al elemento del arreglo
          ADD SI,2         ;Siguiente elemento del arreglo
          LOOP otro        ;Hace n veces (n en CX)
alto:     POP BP           ;Restaura BP
          RET 6            ;Regresa y suma 6 a SP para dejarlo
                           ;apuntando a la cima de la pila
```

Nótese el uso de ***RET*** con un número que debe sumarse a ***SP*** para ajustar a inicio de pila.

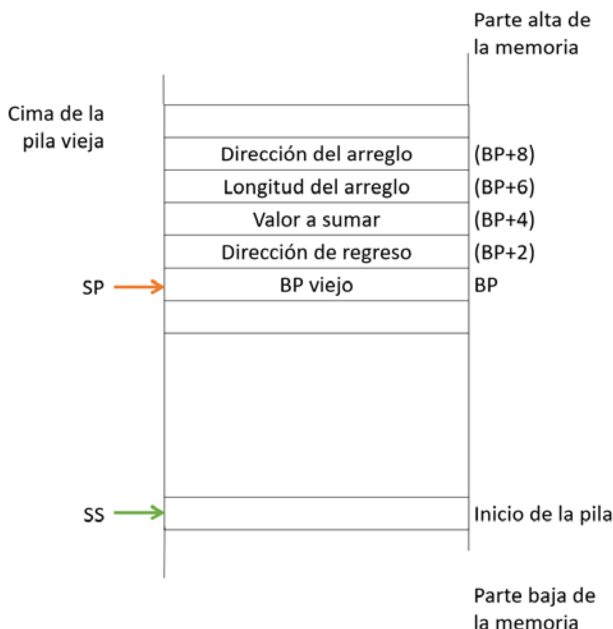


Figura 14.141 Usando el registro BP para acceder a parámetros.

14.10 Interrupciones

Ya hemos estudiado las interrupciones como un mecanismo de los dispositivos externos para informarle a la ***UPC*** que existe actividad externa. La ***UPC*** de la familia 80x86 de Intel detecta este tipo de interrupciones, además de que se pueden generar internamente por medio de instrucciones. Las interrupciones externas entran a la ***UPC*** por medio de dos conexiones: una enmascarable y una no enmascarable. Si la bandera ***IF*** (bandera de interrupción) así lo permite, la interrupción enmascarable es atendida; a la instrucción no enmascarable no es posible deshabilitarla y se usa para eventos catastróficos como: falla de corriente, error en la memoria, saturación de la pila, etc.

Como se lista en la tabla *14.11*, las interrupciones por programa se simulan con las instrucciones ***INT*** (interrumpe) seguido del número de interrupción o ***INTO*** (interrupción y saturación). Las interrupciones que genera el mismo circuito son las de:

- Paso a paso. Interrupción después de cada instrucción causada por fijar la bandera ***TF***.
- Error de división. Causada por saturación de registros.

Una interrupción causa una secuencia de eventos dentro de la ***UPC***:

1. Se termina la instrucción en proceso.
2. Se empujan las banderas a la pila.
3. Se limpia la bandera ***TF*** (trampa) e ***IF*** (interrupción).
4. Se empuja el registro ***CS*** e ***IP*** a la pila.
5. Se carga ***CS*** e ***IP*** del vector de interrupciones multiplicado por 4 (en memorias de 8 bits) para ajustar a la longitud necesaria.
6. Se ejecuta el programa de atención a la interrupción.

Se deben resaltar los siguientes puntos:

a. Ciertos tipos de interrupciones han sido reservadas: la 0 para error de división; la 1 para paso a paso; la 2 para interrupción no enmascarable; la 3 para un punto de interrupción; la 4 para saturación y de la 5 a la 31 para el fabricante.
b. La tabla de interrupciones se localiza en el primer 1K byte de la parte baja de la memoria.
c. La tabla debe ser llenada por el sistema operativo al ser cargado.
d. Se deben salvar los registros que se piensen utilizar en la rutina de interrupción, antes de usarlos, para preservar su contenido.
e. La instrucción ***IRET*** (regreso de interrupción) debe ser la última de la rutina de servicio y es similar a regreso de procedimientos, pero restaura las banderas.

14.11 Instrucciones con Cadenas

Muchas veces es necesario manipular cadenas de información de una u otra forma. El circuito 80x86 nos ofrece una serie de instrucciones (ver tabla *14.13*) que nos facilitan las cosas.

Tabla 14.13	
Instrucciones para Manipular Cadenas	
Mnemónico	**Descripción de la Operación**
MOVSB/MOVSW	Mueve cadena de byte o palabra
CMPSB/CMPSW	Compara cadena de byte o palabra
SCASB/SCASW	Busca cadena de byte o palabra
LODSB/LODSW	Carga cadena de byte o palabra
STOSB/STOSW	Guarda una cadena de byte o palabra

Las instrucciones anteriores se utilizan con los siguientes prefijos:	
REP	Repite
REPE/REPZ	Repite mientras sea Igual/Cero
REPNE/REPNZ	Repite mientras no sea Igual/Cero

Tabla 14.14	
Registros Dedicados para Operaciones con Cadenas (Ver la tabla *14.13*)	
Registro	**Operación**
SI	Índice (desplazamiento) de cadena fuente
DI	Índice (desplazamiento) de cadena destino
ES	Segmento de la cadena destino
CX	Contador de repetición
AL/AX	Valor de búsqueda, destino para *LODS*, fuente para *STOS*
DF	0=autoincrementa *DI* y *SI* 1=auto disminuye *DI* y *SI*
ZF	Indica fin de búsqueda/comparación

En realidad, las cadenas no ahorran mucho, aunque sí hacen más claro el propósito de un programa (que ya es mucho tratándose de ensamblador). En el siguiente ejemplo mostramos dos programas que realizan la misma operación: mover cadenas de caracteres, pero uno usa las instrucciones específicas de cadenas:

```
;Mueve un bloque que comienza en BLOQUE1 a un bloque que
;comienza en BLOQUE2.
;El registro DS es el apuntador al segmento que contiene
;a los bloques. Cada bloque es de 80 palabras (50 hexadecimal)
          LEA SI,bloque1   ;Coloca los apuntadores
          LEA DI,bloque2   ;de los bloques en los registros
                           ;índice SI y DI
          MOV CX,0050h     ;Número de palabras a mover (80)
otro:     MOV AX,[SI]      ;Trae del bloque1
          MOV [DI],AX      ;llevando al bloque2
          ADD SI,2         ;Mueve el apuntador de la
                           ;siguiente palabra
          ADD DI,2
          LOOP otro        ;y ve por otro hasta
                           ;terminar los 80
;Este mismo programa se puede hacer de la siguiente forma :
          PUSH DS          ;Hacer que ES apunte al mismo segmento
          PUSH ES          ;que DS para usar como destino
          LEA SI,bloque1
          LEA DI,bloque2
          MOV CX,0050h     ;Cuenta de 80 palabras
          CLD              ;limpia la bandera de dirección
                           ;para que SI y DI autoincrementen
          REP MOVSW        ;Lo mismo que el bucle anterior
```

```
                              ;pero AX no se afecta
```

Otro ejemplo de programa que busca un byte particular en memoria y lo reemplaza con otro se muestra a continuación:

```
; *************************************************
;Buscar un byte en memoria y reemplazarlo con otro
;Los parámetros que se le pasan en este orden:
;Segmento de la memoria (buffer)
;Desplazamiento de la memoria
;Número de bytes a buscar
;Una palabra cuya parte baja es el carácter a buscar
;y la parte alta es el carácter con el que se le reemplazará
;*************************************************
;
          PUSH BP           ;No es conveniente usar SP
          MOV BP,SP
;
;Poner segmento en ES y desplazamiento en DI
;
          LES DI,[BP+8]
;
;usando una doble palabra en la pila (puede ser necesario
;indicárselo al ensamblador con LES DI,DWORD PTR [BP+8])
;
          MOV CX,[BP+6]     ;longitud a buscar
          MOV AX,[BP+4]     ;AL contiene el carácter a buscar,
                            ; AH carácter a substituir
          CLD               ;limpia la bandera de dirección para
                            ;autoincremento
busca:    REPNE SCASB       ;busca en memoria hasta encontrar AL
          JE intercambia    ;Si se encontró, reemplaza
          JMP listo         ;Si no se encontró termina
;
;Usando DI con un desplazamiento de -1, puesto que SCAS realiza
;un incremento de DI, transfiere un nuevo carácter usando ES como
;segmento (y no DS)
;
intercambia:            MOV ES:[DI-1],AH
          JCXZ listo        ;En caso de que la última
                            ;coincidencia esté también
                            ;al final de la memoria
                            ;busca también esta localidad
          JMP busca
listo:    POP BP            ;Restaura BP
          RET 8             ;Regresa ajustando la pila
```

14.13 Instrucciones para Control del Proceso

Estas instrucciones permiten a los programas controlar varias funciones del circuito, tal como se muestran en la tabla *14.15*.

Tabla 14.5	
Instrucciones de Control de Proceso	
Instrucción	**Acción**
STC	Fija bandera de acarreo
CLC	Limpia bandera de acarreo
CMC	Limpia bandera de complemento

STD	Fija bandera de dirección
CLD	Limpia bandera de dirección
STI	Habilita interrupciones
CLI	Deshabilita interrupciones
HLT	Detiene el procesamiento hasta que se inicialice el circuito o haya una interrupción externa
WAIT	Espera a que test (pata externa física del circuito) esté en 1
ESC	Escapa a procesador externo (coproceso)
LOCK	Bloquea el bus durante la siguiente instrucción
NOP	No hacer nada (para rellenar tiempos)

14.14 El Coprocesamiento

El coprocesar se refiere a delegar responsabilidades a otros procesadores para ejecutar ciertos tipos de instrucciones que son tardadas o que pueden ser ejecutadas en forma más eficiente por otro circuito.

El cálculo de ciertas funciones como cosenos, senos, multiplicaciones y divisiones es complicado y tardado; por lo que actualmente se prefiere que otro circuito lo realice. A estos circuitos se les conoce como coprocesadores matemáticos.

Algunas instrucciones usadas para el coprocesador matemático son:

- **FADD** suma real
- **FADDP** suma y saca de la pila real
- **FIADD** suma entero

El usuario, por medio de un interruptor externo, indica si hay o no un coprocesador disponible.

La mayoría de las máquinas no incluyen coprocesador; el circuito 80486 y 80586 lo incluyen en el mismo circuito.

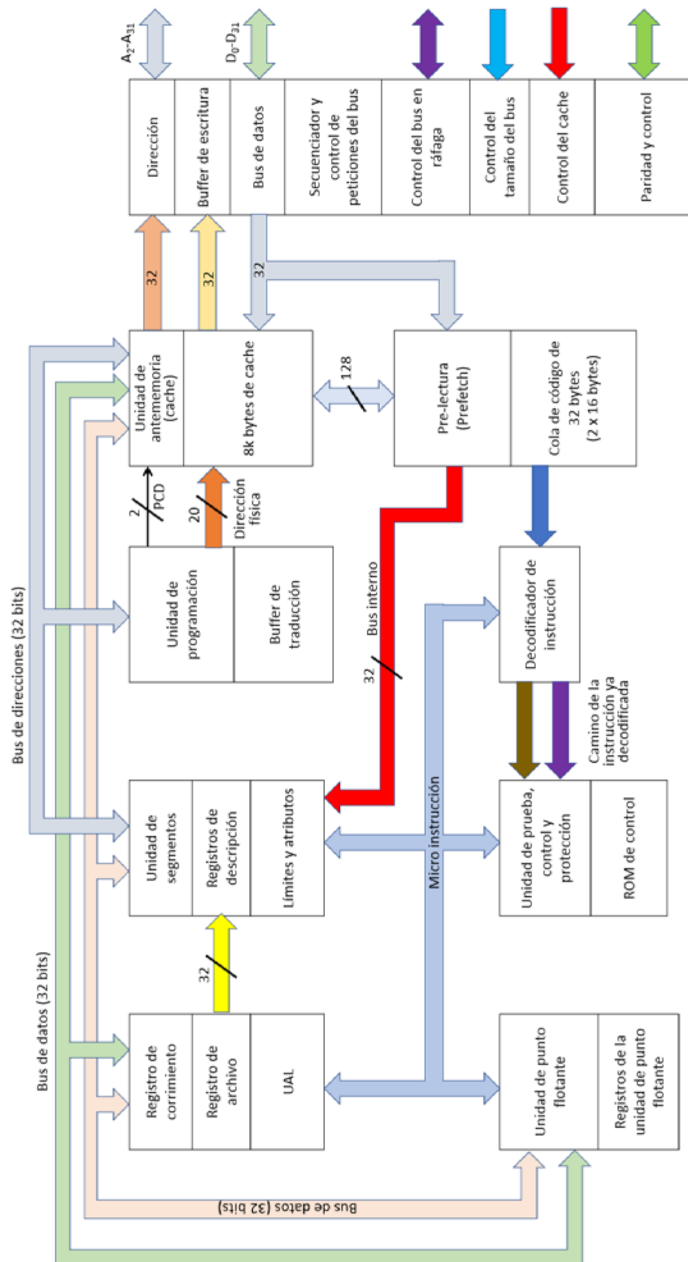


Figura 14.142 Estructura interna del circuito 80486.

14.15 Errores Comunes al Ensamblar un Programa

1. Lógica invertida Ej. ***JC*** (Salta si existe acarreo) cuando se desea saltar si el acarreo=0.
2. Uso equivocado de direccionamiento Ej. *MOV AX, var* cuando lo que queremos hacer es *MOV AX, offset var*.
3. Empujar y no sacar de la pila.

4. Usar un registro para valor temporal y llamar una subrutina que lo use.
5. Olvidar poner en ***CX*** el número de repeticiones cuando se entra a un bucle.
6. Creer que una instrucción afecta a una bandera cuando en realidad no lo hace.
7. Olvidar inicializar un registro.
8. Olvidar el orden destino→fuente.
9. Contar mal los ciclos Ej. del área 100 a 103 hay 4 ciclos, no 3.
10. Incrementar contadores de bucles en lugares equivocados, o no incrementarlos.
11. Olvidar salvar registros y banderas en programas que manejan interrupciones, lo que da por resultado bloques aleatorios.
12. No reservar un área de memoria suficientemente grande para la pila en un programa, pues en el curso de éste pueden ocurrir interrupciones y saturar el área reservada.

14.16 Resumen

La única forma de que un programador pueda sacar provecho de las instrucciones que le ofrece el constructor de un circuito determinado, es conocer a la perfección su funcionamiento y su interrelación con otros componentes dentro del sistema. Para la realización de un programa en lenguaje de máquina se requiere de gran esfuerzo y tiempo.

Aunque el conjunto de instrucciones encontradas en cualquier Unidad de Procesamiento Central típica es muy reducido y cada una de ellas realiza una función muy específica y limitada, la unión de todas ellas es lo que forma los sistemas funcionales de hoy en día. Corresponde al programador de sistemas realizar estas labores y programar las aplicaciones requeridas por otros usuarios y por el propio sistema para que funcione: compiladores, sistemas operativos, interpretes, manejadores, sistemas básicos de entrada y salida, etc.

Se presenta en este capítulo un ejemplo de lenguaje de un procesador muy popular y común en México y en el mundo: la familia 80x86 de Intel que forma la base de las microcomputadoras IBM y compatibles y, desde hace poco, también de las Apple.

14.16.1 Puntos Importantes del Capítulo

- Los registros generales se forman de n bits que componen la palabra de la ***UPC***.
- Algunos registros tienen usos específicos y son usados para ciertas operaciones.
- El registro de banderas determina muchas veces el flujo del programa y las decisiones que deben hacerse dentro del mismo.
- El uso de la memoria segmentada sigue en aumento y se ha relegado la memoria lineal a su uso en pequeños sistemas o dentro de los segmentos.
- Existen muchas formas de realizar un direccionamiento dentro de una instrucción y entre ellas están las de forma inmediata, de registro, indirectas e índices junto con combinaciones y variaciones de las anteriores.
- Las instrucciones son divididas en grupos según sus funciones y cada fabricante las agrupa de distintas formas.
- Cada fabricante agrega o quita instrucciones que, según su experiencia, se requieren o salen sobrando. En general existen dos tendencias dentro del juego de instrucciones:
    a) Que cada instrucción realice el máximo de operaciones (***CISC***).
    b) Que cada instrucción realice sólo lo básico (***RISC***) y sea muy rápida.
- En los diseños modernos se utilizan procesadores auxiliares especializados en ejecutar a máxima velocidad ciertos tipos de instrucciones; son los llamados coprocesadores.

14.17 Problemas

14.1 Usando el ensamblador gratuito de Windows (Debug[52]) o cualquier otro, introduzca a la máquina el siguiente ejemplo y diga qué hace (si no usa Debug deberá corregir cualquier error eventual; vea el apéndice correspondiente para las instrucciones y el uso de este programa):

```
MOV DH,1
MOV AH,2
INT 21
INT 20
```

[52] DEBUG se distribuye gratuitamente sólo con los sistemas operativos Windows de 32 bits. No está disponible en versiones de 64 bits.

Recuerde que si usa el programa DEBUG:

1. Debe empezar en la dirección 100h del segmento en que se encuentre.
2. La dirección absoluta se construye sumando el segmento al desplazamiento ajustado a párrafo: 08F1:0120 es 08F10+0120=09030 y la máxima es FFFF:000F.
3. Para guardar un programa se debe:
   a) Nombrarlo con la opción ***N*** nombre.ext
   b) En ***CX*** poner la parte baja del número de bytes a salvar y en ***BX*** la parte alta.
   c) Usar la opción ***W*** para escribir a disco.

14.2 Realice el mismo proceso del problema *14.1* con:

```
          IN AL,61
          AND AL,FC
cambia:   XOR AL,02
          OUT 61,AL
          MOV CX,0140
aqui:     LOOP aqui
          JMP cambia
```

14.3 La forma general de un programa en ensamblador es la siguiente:

```
;para definir el tamaño de la hoja si se quiere imprimir:
    page 66,132
;nombre del programa con descripción y variables
;autor, fecha, versión
;--------
;definiciones con la instrucción EQU
;--------
datarea segment ;define DS
;datos aquí con DB (datos DB 'hola$')
datarea ends
;--------
prognom segment ;define CS
;--------
main proc far ;parte principal del programa
    assume CS:prognom, DS:datarea
comienzo: ;aquí programa principal
;prepara stack
    PUSH DS
    SUB AX,AX
;prepara DS con segmento
    MOV AX,datarea
    MOV DS,AX
;parte principal programa
    RET   ;regresa a S.O.
main endp
;--------
subr1 proc near
```

Realice los ejemplos de los problemas *14.1* y *14.2* en este formato y ensamble corrigiendo todo posible error que encuentre.

14.4 Como resultado de ensamblar con el formato presentado en el problema *14.3*, tenemos un programa ejecutable desde el intérprete de comandos con la terminación ***COM***. Existen también los programas ejecutables terminados con el sufijo ***EXE*** que presentan varias ventajas:

- Se pueden unir varios de ellos; podemos unir varios módulos programados en distintos lenguajes.
- Usan el segmento correctamente y sacan ventajas de la segmentación.
- Su única desventaja consiste en que tardan más tiempo en cargarse para su ejecución.

Para convertir archivos ***COM*** a ***EXE*** puede seguir este procedimiento general (puede cambiar dependiendo la versión que use de ensamblador):

1) No incluir segmentos de la pila (stack).
2) Poner todos los datos en un segmento de código.
3) Las referencias a ***DS***, ***ES*** y ***SS*** en las líneas de ensamblado ***ASSUME***, se refieren siempre al código.

```
     ASSUME
Programa SEGMENT
principal PROC FAR
ASSUME CS: programa, DS: area_var
ASSUME ES: segx
    .
    .
    .
Programa ENDS
```

4) Se debe incluir una instrucción que determine el origen del programa: ORG 100h
5) Incluir una etiqueta al inicio del programa para que el ensamblador sepa inicializar ***CS*** e ***IP***.
6) No inicializar ***DS***, ***ES*** o ***SS*** en el programa, pues ya apuntan al segmento de código al pasarle el control a éste.
7) Si obtenemos error al ejecutar el programa ***EXE2BIN***, se debe reensamblar con despliegue a la pantalla, y si hay alguna instrucción del tipo _ _ _ _ R, quiere decir que estamos haciendo referencia a algo que está fuera del

segmento de código; esto no es válido para archivos EXE.

8) Al ligar el programa tendremos un error de falta de segmento de pila que hay que ignorar.

Ensamble los programas de los problemas *14.1* y *14.2* como un archivo del tipo EXE.

14.5 Ensamble el siguiente programa como un archivo ejecutable de terminación ***COM*** y ***EXE***:

```
feliz SEGMENT
ASSUME CS:feliz
video     EQU 21h
fin       EQU 20h
          MOV DL,1
          MOV AH,2
          INT video
          INT fin
feliz ENDS
END
```

14.6 Averigüe qué hace la interrupción 20 y 21 en los sistemas ***DOS*** de Microsoft y qué parámetros se les debe pasar.

15 Dispositivos Externos

Los dispositivos externos son todos aquellos no diseñados específicamente para actuar en unión con una ***UPC*** determinada, sin embargo, sin ellos no sería posible la comunicación eficiente de variables de y hacia el sistema de cómputo.

La comunicación entre los dispositivos externos y la ***UPC*** se realiza, como regla general, utilizando los puertos de Entrada/Salida disponibles del sistema.

Toda unión entre el sistema de cómputo y el mundo exterior se realiza por medio de una ***interfaz*** que, como ya se ha dicho, es la frontera o punto de contacto entre dos partes del sistema. En los sistemas digitales se utiliza este término generalmente para referirse al conjunto de señales de conexión que el sistema o cualquiera de sus componentes presenta al mundo externo. Realizar una interfaz significa ligar uno o más componentes o sistemas vía sus respectivos puntos de interfaz, para permitir que la información pueda fluir entre ellos.

Para ligar un dispositivo de ***E/S*** a un sistema de cómputo, se utiliza un circuito de interfaz entre ellos de forma que el circuito realice las funciones de acoplar las señales y lograr la sincronización necesaria para que todo funcione según lo previsto.

El proceso de interfaz tiene dos aspectos bien definidos:

- El electrónico (hardware)
- El programático (software)

Entre los aspectos de la electrónica está el asegurar que las señales eléctricas tengan las mismas características (voltaje, impedancia, forma de onda, etc.), conectar los alambres a las salidas adecuadas y seleccionar los circuitos apropiados de interfaz.

Entre los programáticos contamos con la escritura de los programas adaptados que atiendan a los dispositivos y el control de la transferencia de la información de y hacia los dispositivos de ***E/S***.

Los circuitos de ***E/S*** deben realizar las siguientes funciones:

- Conversión de datos

- Sincronización
- Selección del dispositivo

Algunos de los dispositivos más comunes encontrados en los sistemas que se basan en microcomputadoras son enumerados en la tabla *15.1*.

Tabla 15.1	
Dispositivos de E/S más utilizados	
Tipo	**Dispositivo**
Entrada	Interruptor de prendido/apagado
	Teclado
	Transductor (sensor)
Salida	Diodo emisor de Luz
	Pantalla de visualización
	Motor/actuador eléctrico
	Impresora
Entrada/Salida	Terminal interactiva
	Memoria auxiliar (USB, disco duro o flexible[53])
	Comunicación (bus estándar, módem)

En las siguientes secciones analizaremos cada uno de los componentes más importantes necesarios para formar un sistema de cómputo funcional.

15.1 Fuentes de Poder

Nada de lo hasta ahora analizado puede funcionar sin una fuente de voltaje adecuada. Crear una fuente de voltaje sencilla con una buena regulación ya no es el trabajo que solía ser. Nuevos circuitos simplifican el diseño y lo hacen muy económico.

La fuente básica se forma de un transistor, un puente de rectificación completo (ver capítulo *2*) y dos capacitores que forman la etapa de regulación (ver figura *15.1a*). Esta fuente tan sencilla puede ser funcional para circuitos en los que los requerimientos de regulación (voltaje de salida/voltaje esperado) no sean tan estrictos.

---

[53] En desuso.

Si al circuito anterior le agregamos un diodo Zener, el neto es una regulación superior, pero todavía estamos muy lejos de la regulación requerida en la mayoría de los circuitos digitales. La solución al problema es el uso de circuitos de regulación formados por varias decenas de transistores y resistencias en un arreglo que nos entrega una regulación excelente y poco consumo a bajas corrientes.

Los circuitos reguladores disponibles en forma comercial están formados por una cápsula plástica con tres terminales donde se realizan las conexiones (figura *15.1b*). Los hay de varios voltajes y los de uso más común son los 7805 de 5 voltios y los 7812 de 12 voltios, voltajes comunes en los circuitos digitales electrónicos. La única desventaja es su relativa poca corriente de trabajo (1 amperio) que puede ser mejorada hasta 5 amperes con varios diseños complicados que incluyen arreglos de transistores para amplificar la corriente al mismo voltaje.

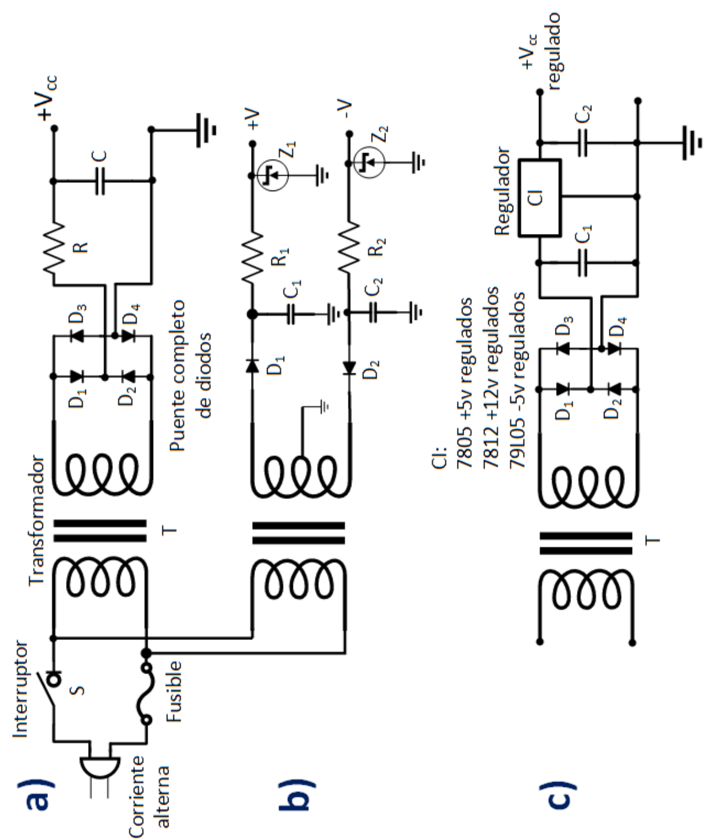


Figura 15.143 Fuente de poder básica.

Cuando el prerrequisito es una gran corriente de salida, esto va acompañado siempre de una gran disipación de energía calorífica que debe ser desviada del circuito para evitar su sobrecalentamiento y fallo eventual. Mientras más frío funcione un circuito, mayor es su vida útil. Para evitar el calor al máximo se utiliza una base metálica (usualmente de aluminio) con disipadores que tratan de entregar a la atmósfera el máximo del calor. Si aun así la temperatura sigue en aumento, se utilizan uno o varios ventiladores.

Los requerimientos de corriente en un sistema de cómputo actual son tan grandes (750w o más[54]) por lo que los circuitos evocados hasta ahora no son prácticos en estos diseños. Han surgido toda una serie de técnicas para mejorar la respuesta de las fuentes de voltaje y aumentar la corriente que pueden entregar mientras se trata que la temperatura de operación sea la más baja posible.

[54] Una microcomputadora moderna consume hasta 96w alcanzando una temperatura de operación de 90º C.

Algunos de los diseños más eficientes son las fuentes de poder con reguladores de interrupción (switching regulator) que, a diferencia de los circuitos anteriores, que son estáticos, dependen de la interrupción constante y exacta de corriente para su funcionamiento y alta eficiencia. Estos tipos de fuentes de poder son más difíciles de diseñar y de construir pues convierten el voltaje de entrada de corriente directa una señal de alta frecuencia, la regula (que es más fácil pues estamos tratando con corriente alterna) y posteriormente la rectifica una vez más para convertirla en corriente directa. Se requiere de un buen blindaje (que es costoso) para evitar que los estados transitorios de corriente pasen a los circuitos digitales. Este diseño es competitivo en precio con los sistemas lineales arriba de 200 Watts. En la figura *15.2* mostramos un circuito de una fuente regulada de interrupción.

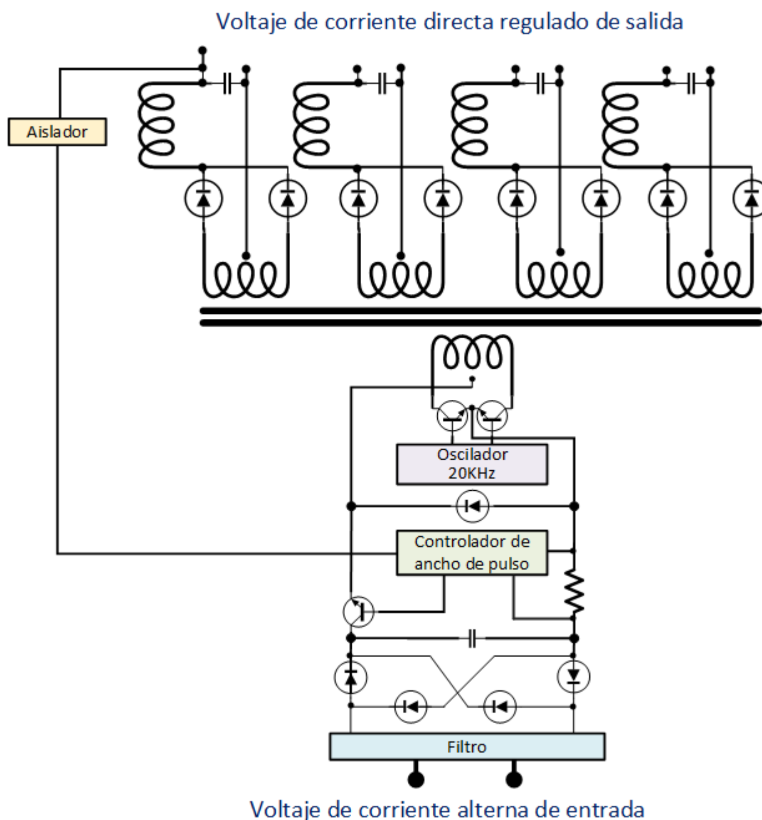


Figura 15.144 Fuente de poder por interrupción.

Otro circuito muy usado es el regulador cuasi-interrumpido (quasi-switching regulator) que sustituye el transformador pesado y caro de un regulador de interrupción con uno más ligero, que lo hace más económico y competitivo, con fuentes lineales de hasta

25 Watts. El espacio que ocupa se reduce también hasta en un tercio de las fuentes convencionales lineales.

15.2 El Reloj

El pulso que controla a todo sistema síncrono digital es el reloj (uno o varios). El diseño del reloj puede ir desde un simple capacitor y resistencia que oscilen hasta un – hasta hace poco, caro, pero exacto – cristal de cuarzo pasando por infinidad de circuitos osciladores menos precisos, pero más económicos.

La elección del reloj depende de la exactitud y estabilidad necesaria, así como de los requerimientos de los circuitos que utilicen la señal de sincronización. En el siguiente capítulo estudiaremos a fondo los distintos circuitos que nos pueden dar una base de tiempo exacta; por ahora nos basta esta breve introducción.

15.3 El Teclado

Entre los dispositivos más sencillos de entrada a un sistema se encuentra el simple interruptor que permite el paso o no de la corriente según su posición. Todo cierre o apertura del interruptor causa un rebote mecánico que, aunque imperceptible a la vista, es fatal para los circuitos digitales. Es por esto que se requiere de la conexión del interruptor junto con la electrónica o lógica que elimine estos efectos.

Una forma común de eliminar estos rebotes no deseados se estudió al introducir los condensadores y los flip–flops, pero otra forma de hacerlo es mediante el uso de un programa que deje pasar un tiempo antes de intentar reconocer qué es lo que el interruptor le está entregando; si un cero o un uno.

Aunque un interruptor es muy útil en ciertos casos, un conjunto de muchos de ellos como es el caso de un ***teclado***, es ciertamente más practico en ciertas situaciones.

Los teclados comunes son una colección de interruptores de prendido y apagado, usualmente del tipo de dos conexiones, cada uno de los cuales se le asigna una función específica tal como introducir un dígito al sistema de cómputo. En la figura *15.3a* presentamos un tablero de interruptores utilizado para introducir números y símbolos matemáticos usados en las calculadoras. Cada una de las 16 teclas es un interruptor que puede estar haciendo o no contacto según se presione o no. Todas las teclas tienen un punto

común por lo que el teclado presenta una interfaz de 17 conexiones eléctricas.

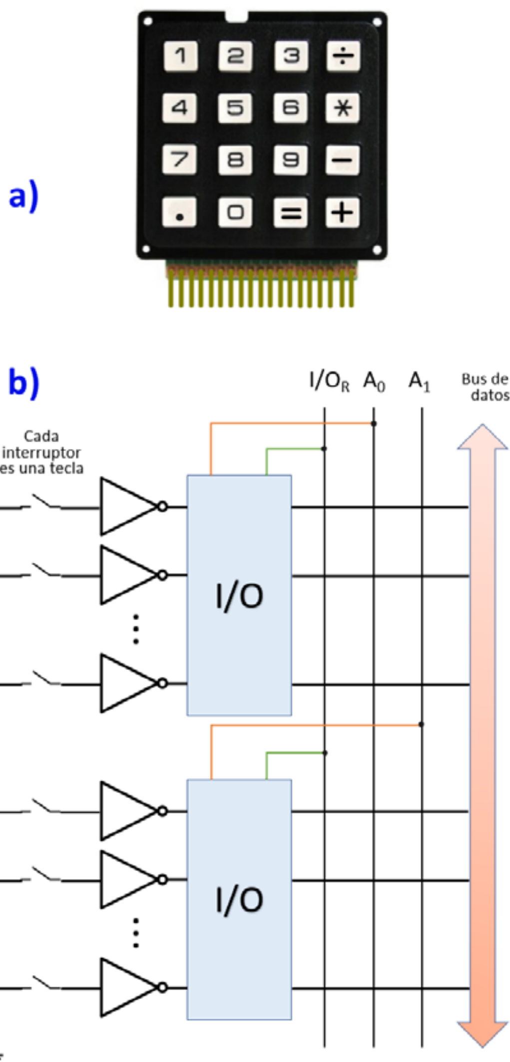


Figura 15.145 Teclado numérico básico.

Una forma directa de realizar la interfaz entre un tipo de teclado y la computadora se muestra en la figura *15.3b*. Dos circuitos de ***E/S*** (***I/O***) sirven de interfaz. Cada una de las teclas se conecta a un bit de la palabra de los circuitos y cada circuito es configurado como un puerto de entrada pasivo (no interrumpe a la ***UPC***) con una dirección única. Las salidas de los circuitos se conectan directamente al bus de datos y se selecciona a los puertos con el bus de control. Se requiere de un programa que controle la transferencia entre el teclado y la computadora. El programa debe realizar las siguientes funciones:

1. Transferir datos entre los dos puertos de entrada usados por el teclado y la ***UPC***.
2. Decodificar los datos de entrada y pasar control a una rutina de acción correspondiente a la tecla presionada.
3. Resolver cualquier ambigüedad causada por el rebote de los contactos o por teclas presionadas simultáneamente.

La simplicidad de la conexión del circuito anterior se ve opacada por la complejidad del programa requerido para controlar el teclado.

Si la forma de interfaz antes descrita trata de extenderse a un teclado con más teclas, por ejemplo, al usado en las computadoras modernas[55], el número de líneas de interfaz y de puertos de ***E/S*** utilizados se vuelve excesivo. Esto se resuelve usando una técnica llamada de los ***Unos Caminando*** (Walking ones). Si observamos la figura *15.4*, el arreglo de las teclas se forma en una matriz cuadrada de 8 x 8 con un interruptor en la unión de cada columna y renglón. Presionar la tecla equivale a realizar el contacto eléctrico entre la correspondiente columna y renglón. Dos puertos de 8 bits son utilizados en el arreglo: uno de ellos configurado como puerto de entrada para los renglones y el otro como puerto de salida para las columnas.

[55] Ciento cinco es la norma.

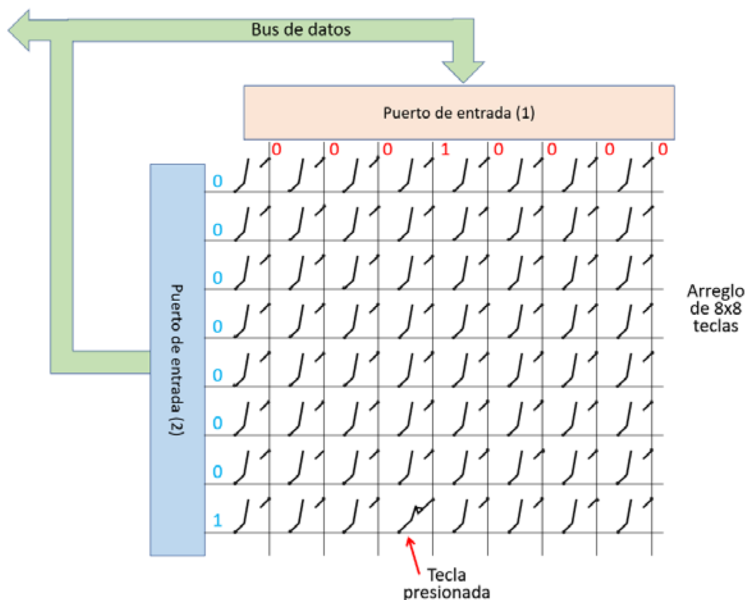


Figura 15.146 Teclado usando la técnica de "unos caminando".

El estado del teclado es determinado por una secuencia de ocho operaciones de escritura y ocho de lectura en el que el patrón escrito contiene un 1 que va caminando en la posición: 1000 0000, 0100 0000, 0010 0000, etc. Si alguna tecla se presiona, el uno pasa al renglón y éste puede leerse a través del puerto de ***E/S*** de lectura identificándose, así, la tecla presionada. Una vez más, el programa debe resolver los conflictos de rebote y de varias teclas presionadas a la vez.

Toda la lógica necesaria para realizar un circuito de unos caminando, puede ser incluida en un paquete de alta integración que recibe el nombre de ***encodificador*** que usualmente incluye también un ***ROM*** para realizar la conversión a ***ASCII*** y los circuitos necesarios para mandar la información en forma serial[56] disminuyendo el número de cables (sólo 5) requeridos en la conexión con la ***UPC***.

15.4 El Monitor

El dispositivo de salida, que es la contraparte de un interruptor, es el diodo emisor de luz (***LED***) que consta de dos estados: apagado (obscuro) o prendido (iluminado). Hemos ya analizado los ***LED*** en el capítulo *2*.

[56] Muchos de ellos también incluyen una interfaz *USB* y transmisión inalámbrica Bluetooth (ver el capítulo *12*).



Peter Mark Roget
(1779-1869)

Médico, físico, matemático, filólogo, teólogo natural y lexicógrafo inglés. Dio en 1824 el primer paso para la explicación del dibujo animado. Intentó explicar el fenómeno de la ilusión óptica que lleva a la persistencia de la imagen en la retina. En virtud de dicho fenómeno, el cerebro las "enlaza" como una sola imagen visual, móvil y continua (ley de Ferry-Porter). Aunque en general 10 fotogramas son suficientes, se usan 25 o más dependiendo el país y uso.

La forma más usual de emplear ***LED*** es como indicadores de estado, pero es también común su uso en sistemas numéricos donde un arreglo de 8 ***LEDs*** que sirven para formar un número. Para controlar a cada uno de los segmentos, se usa un circuito llamado ***BCD a 7 segmentos*** disponible comercialmente en las distintas familias lógicas.

Otro arreglo común es el uso de una matriz de ***LEDs*** que tiene la ventaja de poder desplegar letras y símbolos aparte de los números. Para controlar a una matriz de diodos, es necesario el empleo de circuitos más complejos tales como los mostrados en la figura *15.5* donde se usa una memoria ***ROM*** para generar el carácter a desplegar. Es necesario el uso de un reloj para que las columnas se vayan iluminando en secuencia y el consumo de corriente baje a niveles aceptables; la persistencia de la vista humana da el efecto de que están prendidos constantemente.

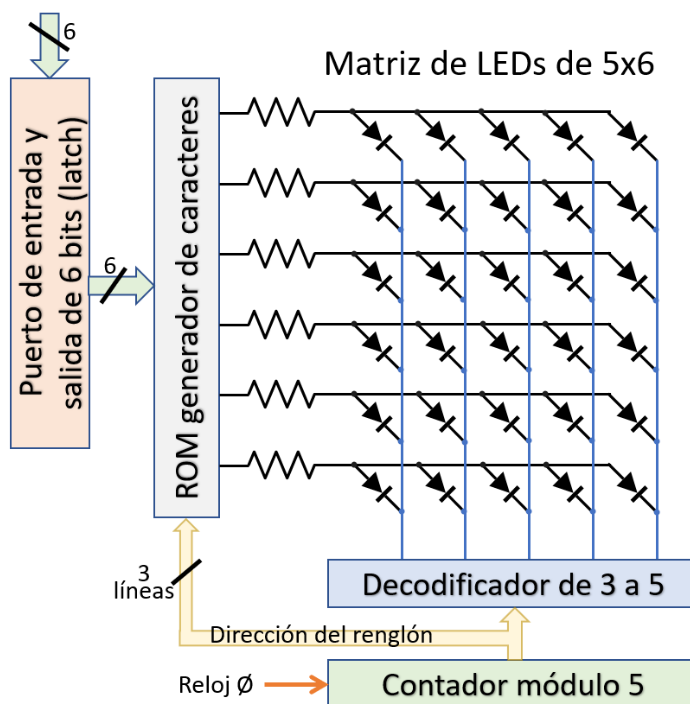


Figura 15.147 Despliegue a base de LEDs.

Cuando se usa más de un despliegue numérico, utilizar la técnica de multiplexar se prefiere en lugar de usar convertidores ***BCD*** para cada uno de ellos. Usando este método sólo uno de los despliegues es activado a la vez por un breve espacio de tiempo (figura *15.6*) y una vez más, se deja que el ojo humano realice el trabajo de armar la figura completa en la retina. A este fenómeno se le

conoce como ***Persistencia de la Visión*** y depende del nivel de iluminación de la pantalla. Es una dependencia de tipo logarítmica y se conoce como ley de Ferry-Porter:

$$(1.17) \qquad 15.1 \qquad f = 37 + 12.6\, log_{10}(B)$$

Donde;

- f es el parpadeo (flicker) del que observa la imagen
- B es el brillo de la pantalla en pies−candelas

Existen ocasiones (la gran mayoría de los casos prácticos de un sistema de cómputo complejo) en los que los despliegues a base de diodos emisores de luz no son suficientes por la gran cantidad de información que se desea desplegar. Cuando estas situaciones suceden, se prefiere el uso de ***monitores*** o ***pantallas*** de video como dispositivos de visualización. Las primeras pantallas de video fueron realizadas con receptores de televisión modificados para saltarse la etapa de detección de la señal de video.

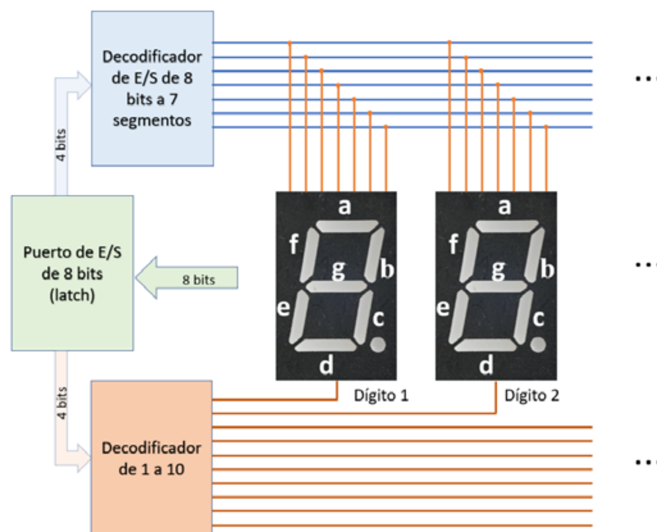


Figura 15.148 Despliegue de 10 dígitos multiplexados.

Estas pantallas se formaban por un tubo de rayos catódicos (***CRT*** o cathode ray tube) que generaban un haz de electrones que incidía en una pantalla frontal cubierta de un material (fósforo) que brillaba al pegar en él los electrones. El punto iluminado se denomina ***Pixel*** (de pix; picture, el; element) y la definición de una pantalla se mide en el número de pixeles horizontales por verticales. Los electrones se guiaban por la electrónica del dispositivo en líneas horizontales hasta llegar al final de la pantalla y luego,

rápidamente, se pasaban al siguiente renglón (flyback) para formar la siguiente línea de puntos luminosos. El movimiento horizontal del haz se llama barrido horizontal y el vertical, barrido vertical. Cuando se llega al final de la pantalla, se regresa al principio para generar el primer renglón de la pantalla (vertical flyback).

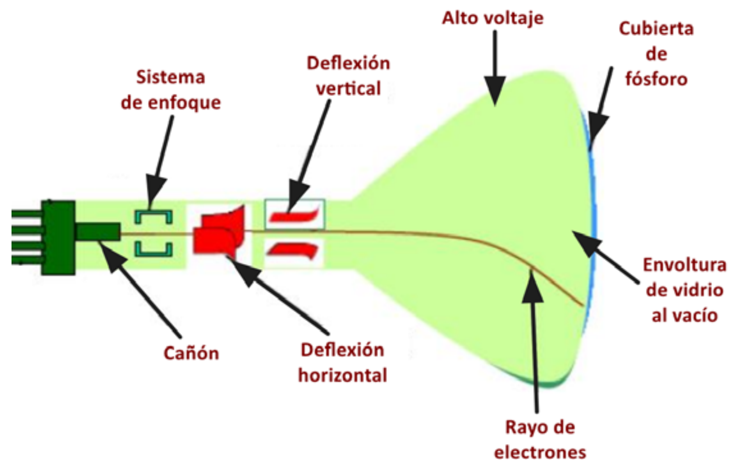


Figura 15.149 Pantalla de tubo de rayos catódicos monocromático.

En un principio los caracteres a desplegar se guardaban en una parte de la memoria principal donde la ***UPC*** podía modificarla por medio de los programas del usuario. Era necesario, entonces, transferir de forma rápida y eficiente este contenido a la lógica del monitor para su despliegue. Para realizar esto se utilizaba un Acceso Directo a Memoria (***DMA***) que llevaba los datos de la memoria principal pasando por un ***ROM*** generador de caracteres al video sin intervención directa de la ***UPC***.

Las funciones complejas de la conversión de caracteres a puntos en la pantalla se dejaban a un circuito llamado controlador de ***CRT*** o ***CRTC*** (cathode ray tube controler)

Este diseño de usar una parte de la memoria principal para el despliegue del video ya no resulta práctico debido a la alta resolución de las pantallas modernas, por lo que se prefiere dejar esto a un circuito especializado residente en la propia tarjeta gráfica. Dicha tarjeta cuenta con su propio ***CPU*** y ***DMA****,* así como con su memoria que comunica con la ***CPU*** principal, con lo que se ha dado por llamar el ***Puente Norte***.

Cuando se requiere de color, se usan tres haces de electrones cada uno de ellos incidiendo en tres puntos distintos de la pantalla que

tienen los colores Rojo, Verde y Azul; según su intensidad relativa nos permiten generar todos los demás colores (ver figura *15.8*).

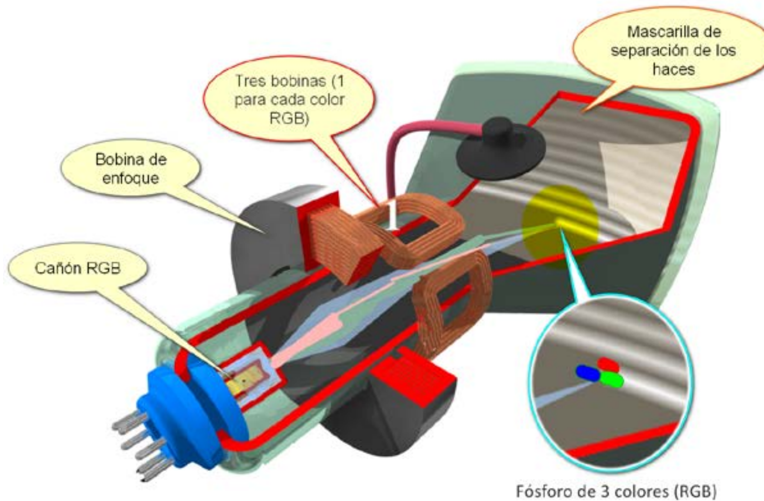


Figura 15.150 Monitor CRT de color.

Es deseable diseñar una pantalla portátil que cumpla entre otros con las siguientes características:

- Consumir poca energía
- Ser casi plana
- Ser ligera
- Ser barata

Utilizando una tecnología llamada de despliegue de cristal líquido o ***LCD***[57] (Liquid Crystal Display) es posible realizar pantallas casi planas. Cada pixel (punto de la imagen) se construye con dos capas de vidrio que encierran electrodos transparentes y cristal líquido. Un despliegue ***LCD*** no produce luz y requiere iluminación por la parte de atrás del sándwich de vidrio o usar un espejo que refleje la luz ambiental para iluminarse.

El líquido entre los vidrios es generalmente transparente, pero al pasar una corriente por los electrodos se produce un fuerte campo magnético que alinea las moléculas del líquido en un sentido (polariza la luz) impidiendo el paso de la luz. Esta polarización se muestra como negro al no permitir el paso de la luz de una pantalla luminosa en el fondo del arreglo (ver figura *15.9*).



Friedrich Reinitzer (1858-1927)

Botanista y químico austriaco. En 1888 descubrió la naturaleza cristalina líquida del colesterol extraído de las zanahorias, lo que se conocería más tarde como cristales líquidos, publicando sus hallazgos en una reunión de la Sociedad Química de Viena.

[57] Las pantallas de ***LEDs*** han reemplazado tanto a las ***LCD*** como a las ***CRT*** por su eficiencia y bajo consumo. En esencia los monitores ***LED*** son similares a los de ***LCD*** pero en los de ***LED*** se sustituye la luz de polarización con la del propio ***LED***.

La misma teoría del color explicada anteriormente en los monitores ***CRT*** se usa en las pantallas ***LCD*** de color usando tres pixeles de cristal líquido contaminado con elementos que den el colorido adecuado[58].

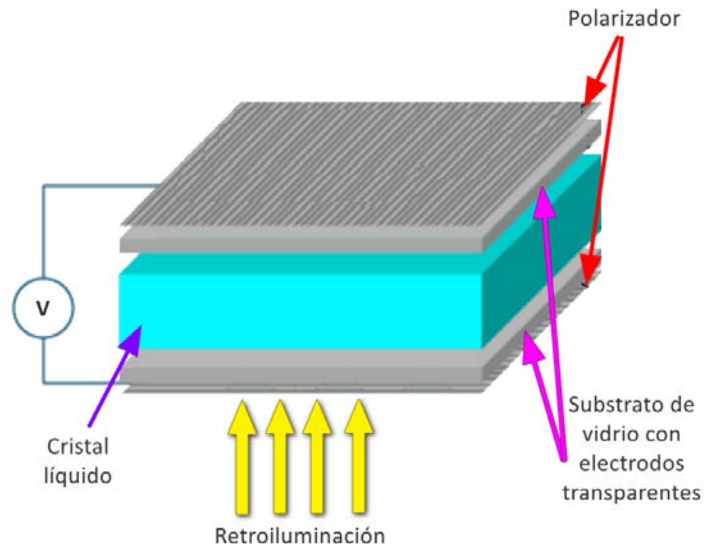


Figura 15.151 Un pixel en una pantalla de cristal de cuarzo líquido o LCD.

Se han desarrollado otras técnicas de despliegue de información sin tanto éxito como las aquí descritas.

15.4.1 Pantallas táctiles

Con el fin de facilitar la vida al usuario y evitar el uso de varios otros dispositivos auxiliares (ratón, joystick y otros) desde su invención en 1971 por George Samuel Hurst se han diseñado y comercializado varios tipos de pantallas táctiles con mayor o menor éxito.

Existen o existieron varios tipos de estas pantallas que permiten su manipulación ya sea con el dedo o con un estilete especial:

- Infrarrojos. Una malla de emisores y captores infrarrojos detectan dónde se coloca el dedo que interrumpe el haz. El resultado se interpreta y envía para su ulterior proceso.
- Resistivas. Compuesta de un vidrio recubierto con varias capas. Las más importantes entre ellas son dos capas plásticas conductivas separadas por un pequeño espacio. La

[58] Considere que un monitor de color con una resolución de 1024x768 pixeles tiene 1024x768x3=2,359,296 elementos de cristal líquido.

superior es la que el usuario toca y la inferior es una capa resistiva. Se aplica voltaje en ambas. El punto en el que el usuario hace contacto se conecta y es detectado e interpretado por la electrónica de la pantalla.

- Capacitiva. Basadas en sensores capacitivos. Consisten en una capa de aislamiento eléctrico, como el cristal, recubierto con un conductor transparente. Siendo el cuerpo humano un conductor eléctrico, el contacto con la superficie de la pantalla genera una distorsión del campo electrostático de la pantalla, que se mide a través del cambio en la capacitancia. Se usan distintas tecnologías para determinar en qué posición de la pantalla se hizo el toque. La posición se envía al controlador, para su procesamiento.
- Onda acústica. Utiliza ondas ultrasónicas que pasan sobre el panel de la pantalla táctil. Cuando se toca el panel, se absorbe una parte de la onda. Este cambio en las ondas ultrasónicas registra la posición del evento táctil y envía esta información al controlador, para su procesamiento.

La figura *15.10* muestra las pantallas táctiles infrarrojas, resistivas y capacitivas.

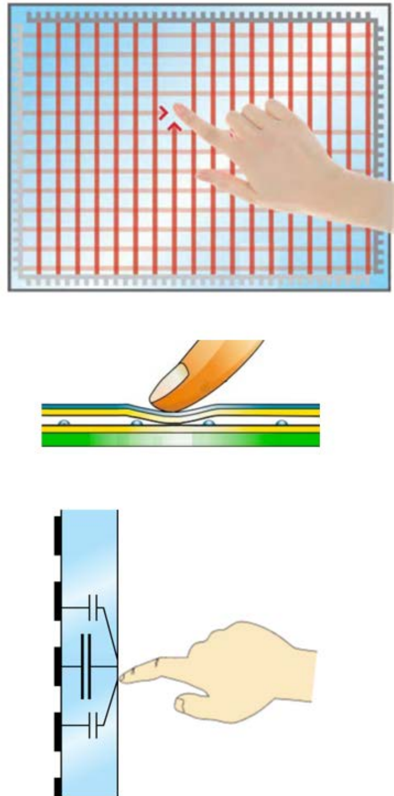


Figura 15.152 Pantallas táctiles.

15.5 Dispositivos de Almacenaje

Casi de nada sirve un computador de uso general si no contamos con un área de almacenamiento temporal de datos de gran volumen y relativamente alta velocidad. Los programas, el sistema operativo y los datos tienen que ser guardados en algún área de memoria principal (***ROM***) pero cuando su tamaño crece, esto se vuelve impráctico; es entonces cuando la utilización de un dispositivo de almacenamiento es aconsejable.

15.5.1 Cintas

Entre los primeros dispositivos de almacenamiento masivo que sustituyeron a la cinta de papel y tarjetas perforadas de forma definitiva y rápida se encuentran las cintas magnéticas usadas casi exclusivamente para respaldos de información masivos que no se requiere que estén en línea (disponibles en todo momento). La cinta magnética[59] es una memoria secuencial donde se almacena información en campos magnéticos grabados en una cinta de

[59] En desuso y usada por muy pocas compañías.

plástico cubierta de un óxido capaz de retener por tiempo finito (más de 10 años) la magnetización a la que es expuesta.

15.5.2 Disco Magnético

El desarrollo de los discos flexibles magnéticos (floppy[60]) relegó casi inmediatamente a la cinta magnética como un dispositivo de respaldo fuera de línea. Las ventajas ofrecidas por el nuevo medio superaron en todo a las cintas:

- Tiempo de acceso mucho menor.
- Costo más accesible.
- Intercambio rápido y eficiente del medio.
- Dispositivo controlador más económico para capacidades similares.
- Acceso aleatorio en lugar de secuencial.
- Mejor manejo del medio para su envío, almacenaje, etc.

El disco flexible es sencillamente un disco maleable de material plástico cubierto de un material magnético y dividido en pistas y sectores donde la información era almacenada. Proveo durante años un método de almacenamiento muy barato y capacidades de medias a altas (de 160K bytes a 2M bytes). Las unidades de 5 1/4" reemplazaron rápidamente a las de 8" y a su vez estas fueron reemplazadas por las de 3 1/2".

El disco flexible se forma de una cubierta externa de cartón flexible (plástico rígido en el caso de los discos más pequeños) y un disco que se hace girar a 360 RPM en su interior. Las cabezas de grabación pueden estar por uno de los lados o por los dos y grabar en densidad sencilla o doble. El disco tiene una perforación central que sirve para hacerlo girar y de uno o más pequeños agujeros en su periferia que sirven para marcar los sectores.

Para reconocer los sectores se usan dos técnicas:

- Sector duro o fijo. Cada sector es marcado por un agujero que se reconoce por un dispositivo óptico (***LED***) y un fototransistor. Quedaron rápidamente en desuso sustituidos por la técnica del sector suave.
- Sector suave. Se marca sólo el inicio del primer sector con una perforación y los demás son reconocidos por medio

[60] En desuso y citado aquí sólo como referencia.

de programa. Es necesario inicializar los sectores en un disco nuevo para que sean reconocidos por la lectora de discos.

La lógica de la controladora lectora de discos flexibles ***FDC*** (Flexible Disk Controller) genera hacia la ***UPC***:

- Petición de interrupción.
- Petición de transmisión.
- 8 o más líneas de datos.

La controladora recibe de la ***UPC*** las siguientes señales:

- 8 o más líneas de datos.
- Reloj.
- Señal de lectura/escritura.
- Pulsos de selección (conectados al bus de direcciones).
- Señal de recepción de transmisión.
- Aceptación de interrupción.

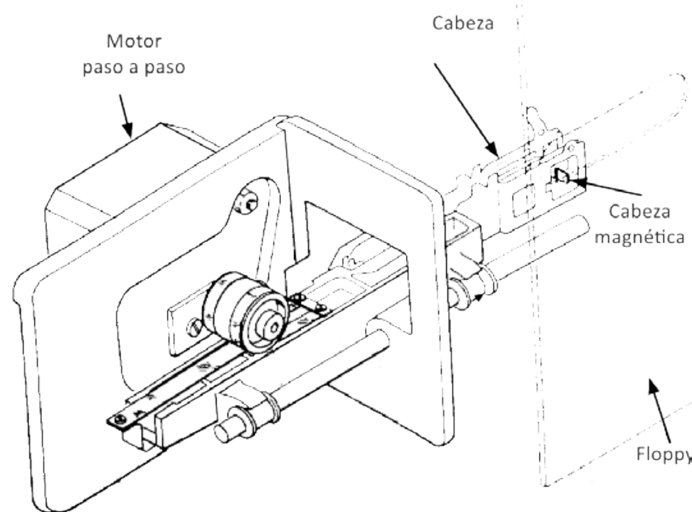


Figura 15.153 Cabeza lectora de disco flexible (floppy).

15.5.3 Disco Duro

Aunque un disco flexible era un medio ideal para almacenamiento de información en bajas cantidades, era inútil en sistemas donde se requerían grandes cantidades de información y/o velocidad. Con el aumento de la complejidad de los sistemas operativos, programas y datos de los usuarios dicha tecnología se substituyó rápidamente con los discos duros. Con los precios cayendo constantemente, el uso del disco duro se generalizó y, en sistemas de

pequeños a medianos, se usa ahora tanto como medio de almacenamiento interno como externo (removible o fuera de línea).

Un disco duro sigue casi el mismo principio que el disco flexible pero ahora el material de grabación se forma por un disco de metal rígido como el aluminio (el vidrio o plástico rígido se emplean en algunos). Los discos rígidos se apilan en un eje, cada disco con su propia cabeza de lectura/escritura y se sellan en un contenedor metálico al alto vacío.

Se requiere usar el vacío en el mecanismo pues la velocidad de giro va paulatinamente en aumento pasando de 3,600 a 7,600, 9,600, 10,000 y hasta 15,000 rpm, la cabeza no hace contacto con el disco, sino que flota a unas cuantas micras de la superficie magnética. Todo contacto con una partícula contaminante, por pequeña que esta sea, sería fatal por las grandes cantidades de información almacenadas.

La velocidad de acceso en lectura y escritura disminuye notablemente, así como el precio por GB almacenado.

Las pistas creadas por la rotación del medio con respecto al material magnético, son ahora cilindros porque no se usa un sólo disco duro.



Figura 15.154 Lectora de disco duro.

15.5.4 Disco de Estado Sólido o SSD

Como ya hemos explicado anteriormente, un disco duro se compone de varios discos metálicos magnéticos giratorios que almacenan datos y varios cabezales de lectura / escritura en brazos mecánicos que se mueven sobre la superficie de dichos discos. Para leer o escribir datos en un sector específico de un disco, el cabezal tiene que moverse a la posición apropiada y luego esperar a que el sector pase por debajo mientras gira el disco.

Este modo de funcionamiento presenta dos fuentes obvias de retraso:

- Se necesita tiempo para que la cabeza se mueva a la posición correcta, lo que se conoce como tiempo de búsqueda.
- Hay una demora mientras el cabezal espera a que llegue la parte correcta del disco, lo que se conoce como latencia de rotación o simplemente latencia.

El tiempo de búsqueda depende de dónde esté el cabezal al inicio de una operación y hacia dónde debe moverse; la latencia depende de la posición del disco en su ciclo, por lo que para un ***HDD*** determinado es normal hablar de tiempo promedio de búsqueda y latencia media.

Otra fuente potencial de retraso es la interfaz ***HDD*** a través de la cual los datos de las unidades se transmiten a una computadora o sistema de almacenamiento conectado. Pero las interfaces comunes de estos discos como la ***IDE*** (***ATA***), ***SATA*** y ***SAS*** se han diseñado teniendo en cuenta el rendimiento de dichos discos, y éstas no suelen ser un factor limitante para las velocidades de lectura y escritura.

Cuando se trata de medir la velocidad de un disco duro, hay cuatro de ellas que son importantes:

- Velocidad de lectura secuencial: lectura de un gran bloque de datos contiguos.
- Velocidad de escritura secuencial: escritura de un gran bloque de datos contiguos.
- Velocidad de lectura aleatoria: lectura de datos esparcidos por todo el disco.

- Velocidad de escritura aleatoria. Las velocidades aleatorias son generalmente mucho más bajas que las velocidades secuenciales debido a la cantidad de latencia de búsqueda y la rotación involucrada.

15.5.4.1 Cómo funciona una SSD

La forma en que funciona una unidad ***SSD*** (Solid State Drive) es completamente diferente a una unidad ***HDD*** (Hard Disk Drive). Mientras las unidades ***HDD*** usan magnetismo, las ***SSD*** utilizan un medio de almacenamiento de estado sólido, típicamente compuertas tipo ***NAND*** (a menudo conocida como memoria flash). Los datos persisten aún sin fuente eléctrica externa. Los datos se escriben o se leen desde la ***NAND*** mediante un controlador, que efectivamente es el cerebro del dispositivo.


Fujio Masuoka
(1943-)

Ingeniero Eléctrico japonés inventor de la memoria flash para Toshiba. Usada actualmente como reemplazo de los discos duros magnéticos y casi todo otro medio de respaldo externo. Investiga ahora el desarrollo de transistores tridimensionales. Se le llama memoria “flash” porque se puede borrar y reprogramar “instantáneamente” – tan rápido como el flash de una cámara fotográfica.

Con un ***SSD*** no hay tiempo de búsqueda variable ni latencia de rotación, ya que se puede acceder a cada parte del ***SSD*** en la misma cantidad de tiempo. Pero las velocidades de lectura y escritura de ***SSD*** son asimétricas: las lecturas de datos son muy rápidas, pero las velocidades de escritura de ***SSD*** son algo más lentas.

Esto se debe a que el almacenamiento ***SSD*** se compone de celdas ***NAND*** individuales que pueden almacenar uno (o sólo unos pocos) bits de datos; los grupos de celdas se organizan en páginas. Finalmente, los grupos de páginas se organizan en bloques.

El problema es que los datos no se pueden escribir en una celda a menos que se borren primero, eliminando cualquier información existente, y aunque los datos se pueden escribir una página a la vez, sólo se pueden borrar en bloques completos a la vez. Esto significa que para escribir un sólo bit de datos en una celda es necesario copiar todas las páginas del bloque que contiene esa celda en un área de esperar; borrar todo el bloque y luego volver a escribir todas las páginas y el nuevo bit de datos. al bloque borrado.

El mismo diseño del estado sólido del almacenamiento ***SSD*** le da una gran ventaja de velocidad sobre el diseño mecánico del almacenamiento ***HDD***, con los retrasos inherentes que conlleva. Esto hace que, aunque se tenga que borrar antes de escribir, la velocidad de un ***SSD*** es netamente superior a la de un ***HDD***.

Por supuesto, qué tanto más rápido depende de qué ***SSD*** y ***HDD*** se compare y qué se está comparando exactamente. Una

comparación de velocidades de ***SSD*** revelará que existe una amplia variación entre las velocidades de ***SSD***.

Pero, para tener una idea de la diferencia de rendimiento que podría mostrar una comparación de velocidad de ***SSD*** vs un ***HDD***, se podría decir que un ***SSD*** estándar puede leer datos secuenciales a una velocidad de aproximadamente 550 megabytes por segundo (*MBps*) y escribirlos a 520MBps. Por el contrario, un ***HDD*** rápido puede realizar lecturas y escrituras secuenciales a sólo 125 *MBps*.

Esto muestra que la diferencia entre el rendimiento de ***SSD*** y ***HDD*** es significativa. La respuesta a la pregunta de cuánto más rápido es un ***SSD*** en comparación con un ***HDD***: aproximadamente cuatro veces más rápido cuando se trata de velocidad de lectura ***SSD*** frente a un ***HDD***, y un poco menos cuando se compara la velocidad de escritura ***SSD*** frente a ***HDD***.

Otra gran ventaja es que los ***SSD*** se han diseñado para ser reemplazos directos de los ***HDD***, y esto significa que a menudo se fabrican con las mismas interfaces que los ***HDD***. Dichas interfaces se han optimizado específicamente para dispositivos de almacenamiento ***HDD***, pero no son óptimas para ***SSD***. Esto hace que los discos ***SSD*** (o memoria ***Flash*** como también se les conoce) no logren su máximo potencial.

Con sólo aumentar en número de líneas del bus de interconexión del ***SSD*** a los chips ***NAND*** se puede fácilmente obtener rendimientos máximos de 3000MB/s.

Otra forma en que se puede aumentar la velocidad de ***SSD*** es usar ***NAND*** más rápidas. La ***NAND*** estándar que se usa en las ***SSD*** es, en efecto, una tecnología plana, y los algoritmos de corrección de errores que se usan para mitigar la corrupción de datos causada por la interferencia entre celdas muy compactas, reducen el rendimiento. Existe una nueva tecnología de chips flash que utiliza celdas de memoria en capas múltiples (conocidas como ***3D NAND***) y esto ofrece el potencial para un rendimiento de lectura y escritura ***SSD*** mucho más rápido. Esto se debe a que ya no es necesario ejecutar estos algoritmos en ***3D NAND***.

La diferencia entre las velocidades de lectura de ***SSD*** y ***HDD*** se puede mejorar aún más alejándose por completo del diseño con compuertas ***NAND*** y, en su lugar, utilizando ***SSD*** que están

equipados con un nuevo medio de almacenamiento llamado *3D XPoint*, desarrollado conjuntamente por Intel y Micron.

Lo que limita la velocidad de un disco duro es el tiempo de búsqueda (el retraso a medida que el cabezal de lectura/escritura se mueve a su nueva posición) y la latencia (mientras el disco duro espera a que la parte requerida del disco gire a su posición debajo del cabezal), como se explicó anteriormente.

Por lo tanto, al reducir estos dos factores, se puede reducir la diferencia entre el rendimiento de un ***SSD*** y un ***HDD***.

La forma de reducir la latencia en un ***HDD*** es relativamente simple: aumentar la velocidad de rotación de los discos reducirá la latencia y, por esa razón, los discos duros de alto rendimiento giran a 15,000 rpm en lugar de las 7,200 rpm ya estándares en la industria, hoy en día.

Girar un ***HDD*** más allá de 15,000 rpm daría como resultado mayor reducción en la latencia, pero por razones prácticas, esto es difícil de lograr: cuanto más rápido giren los discos, menos estables son. Los discos que giran más rápido también consumen mucha más energía, vibran más y son más ruidosos. Esos problemas se han mitigado en parte llenando los ***HDD*** con helio, pero por el momento parece ser que el límite máximo es de 15,000 rpm.

Es posible reducir aún más los tiempos de búsqueda para aumentar el rendimiento de un ***HDD***, y esto se logra comúnmente usando un truco llamado ***recorrido corto***. Esto implica que en lugar de usar toda la platina del disco duro, se usa sólo una parte de su capacidad total, por ejemplo, usando sólo el 10% más externo de cada disco que lo conforma. Al hacer esto, el cabezal de lectura/escritura sólo tiene que cubrir una distancia de una décima parte y simular como si todo el plato estuviese en uso y, en promedio, estará mucho más cerca de donde necesita moverse para cada operación de lectura o escritura.

La desventaja de los recorridos cortos es que, aunque habrá un aumento significativo en la velocidad del disco duro, es extremadamente ineficaz porque sólo se puede utilizar una pequeña parte de la capacidad de almacenamiento total del disco duro, aunque el consumo de energía no se modifique.

Tabla 15.2		
Tabla Comparativa SSD vs HDD		
Ventaja	**SSD**	**HDD**
Capacidad	Media–baja	Media–alta
Consumo	Muy bajo	Bajo–alto
Coste por Gb	Medio–alto	Bajo–medio
Ruido	No hay	Bajo–alto
Vibraciones	No hay	Bajo–alto
Fragmentación	No hay	Medio–alto
Durabilidad	Bajo–medio	Medio–alto
Tiempo de arranque	Bajo–extremadamente bajo	Medio–alto
Transferencia de datos	Media–muy alta	Baja–media
Magnetismo	No se ve afectado	Se pueden perder datos

15.5.5 Disco Compacto

La necesidad de un dispositivo de almacenamiento masivo, rápido y económico llevó al uso de la tecnología de los discos compactos (compact disk o ***CD***) en la computación.

Un disco compacto almacena desde alrededor de 600M Bytes de información hasta 8.5GB; es un medio confiable, seguro y barato de guardar información. Puede usarse tanto en su forma de lectura como de lectura y escritura (en forma limitada).

La tecnología fue desarrollada en 1976 por las compañías Philips de Holanda y Sony de Japón. Pero fue hasta 1983 que los primeros dispositivos aparecieron comercialmente.

Un disco estándar de ***CD–ROM*** mide 120 mm de diámetro, tiene un agujero en el centro de 15 mm y un espesor de 1.2mm. A diferencia de los discos flexibles y duros que tienen muchas pistas concéntricas divididas en sectores, el disco compacto tiene una sola en espiral, tal como los viejos discos de acetato usados para la música. La pista se divide en sectores de igual longitud llamados bloques; en una pulgada hay alrededor de 16,000 vueltas de la misma pista.

Los datos se graban en los bloques en forma de pequeñas hendiduras (pits) y valles (lands) y una cabeza lectora de rayo láser las

interpreta; los valles reflejan la luz hacia la cabeza mientras que las hendiduras dispersan la luz.

El disco es de un substrato trasparente que luego es cubierto con una superficie de aluminio altamente reflejante y una capa de laca protectora.

Existen discos compactos que permiten la escritura una sola vez (***CD−RCD−R***, llamados también ***CD−WOCD−WO***) para luego volverse discos de sólo lectura; cualquier lectora estándar puede leer este tipo de discos, pero se requiere de un dispositivo especial para realizar la grabación inicial y única.

La información del disco debe pasar a velocidad constante sobre la cabeza lectora para poder ser interpretada correctamente. Como el disco se forma de una pista espiral en donde los sectores son de la misma longitud, el mecanismo de giro debe aumentar su velocidad cuando se trata de leer datos en el centro del disco y disminuirla cuando se lee de los sectores próximos al borde externo del disco. Esta variación de la velocidad no permite que la velocidad de transferencia de los datos sea alta y el tiempo de acceso no es menor de 400ms lo que los hace lentos con respecto a un disco duro que está abajo de los 10ms.

El polvo puede causar un caos en la lectura de la información por lo que muchos dispositivos tienen un sistema de doble puerta para evitar al máximo la entrada del polvo al mecanismo de lectura. Otros sistemas incorporan un soplador de polvo en la cabeza que es accionado cada vez que se inserta un disco compacto.

Conforme el precio de las lectoras ha bajado, su popularidad ha ido en aumento; nuevas tecnologías emergen constante y actualmente es posible mezclar texto, video y sonido en el mismo medio; por lo que es la forma preferida de distribuir multimedia y aplicaciones en las que se almacena gran cantidad de información (enciclopedias, dibujos, manuales, revistas, etc.).

15.6 Impresoras

Uno de los dispositivos externos más prácticos, convenientes y comunes es la impresora. De nada sirve tener toda la información requerida procesada si de alguna forma no podemos visualizarla fácilmente. Uno de los medios preferidos por muchas personas para ello es, y ha sido, un registro impreso.

Las impresoras son de los dispositivos externos que más han evolucionado en los últimos años y cada día se mejora aún más su calidad, rapidez, nivel de ruido, tamaño y precio. Muchas tecnologías han casi desaparecido o han sido confinadas a aplicaciones muy específicas o especiales. Un ejemplo de obsolescencia son las impresoras térmicas que encuentra su uso actual en los faxes de bajo costo y terminales portátiles o las impresoras de banda que son usadas en aplicaciones donde se requiere de gran rapidez.

Casi todas las tecnologías están siendo substituidas de una u otra forma por las impresoras de rayo láser o de inyección de tinta que ofrecen rapidez, funcionamiento silencioso, buena calidad de impresión, color y una gran variedad de tipografía (formas de letras).

15.6.1 Máquinas de Escribir

En un principio, la mayoría de las impresoras eran máquinas de escribir adaptadas para esta función. La información se enviaba un carácter a la vez que se interpretaba y servía para accionar un solenoide que impulsaba la letra correspondiente. Los caracteres eran enviados uno a la vez usando para esto un código preestablecido (usualmente el ***ASCII*** de 8 bits) y enviando los bits todos a la vez, esto es usando ***n*** cables para enviar la información en paralelo[61].

Si la información es enviada un bit a la vez se habla de comunicación en serie y es una forma conveniente de comunicación pues sólo es necesario un número reducido de cables.

Una adaptación más de las máquinas de escribir consistió en colocar los caracteres realzados en una rueda usando un sólo mecanismo impulsor consistente en un martillo que se accionaba al pasar la letra sobre él. Por la forma de la rueda se le nombró margarita y tiene la ventaja de ser intercambiable permitiendo el uso de tipografía variada y de muchos tamaños.

15.6.2 Máquinas Impresoras de Bandas

Otro tipo de impresora que estuvo muy en uso y aún hoy en día es utilizada frecuentemente, es aquella en la que los caracteres se colocan realzados en una banda que gira a gran velocidad en unos ejes. El carácter es impulsado por un martillo en el momento justo para incidir sobre carbón y éste sobre el papel. La velocidad de

[61] Interfaz en desuso

operación puede ser muy alta, aunque el mecanismo y el control electrónico es costoso.

15.6.3 Matriz de Puntos

Para tratar de lograr aún más velocidad se ideó un mecanismo en la que la cabeza recorre el papel de lado a lado y en la cabeza se encuentran varias agujas (9) que forman los caracteres sobre la superficie de impresión. Hoy en día estas impresoras están en desuso, pero encuentran su aplicación frecuente en impresión de multiformas o cheques de alta seguridad.

Los caracteres formados son de baja calidad, pero puede aumentar si se usan más agujas en la cabeza (12 o 24) o si se traslapan los puntos con un pequeño desfase en el viaje de vuelta del mecanismo impresor. Esto último hace que la velocidad de impresión baje considerablemente, aunque la calidad sube mucho.

La velocidad de impresión es de media a alta y puede ser aumentada con varias técnicas, siendo la más común aquellas en las que los caracteres se forman tanto en el viaje de ida como en el de vuelta de la cabeza. Otra forma consiste en aumentar el número de cabezas, pero esto hace mucho más costoso el mecanismo y las fallas son más frecuentes.

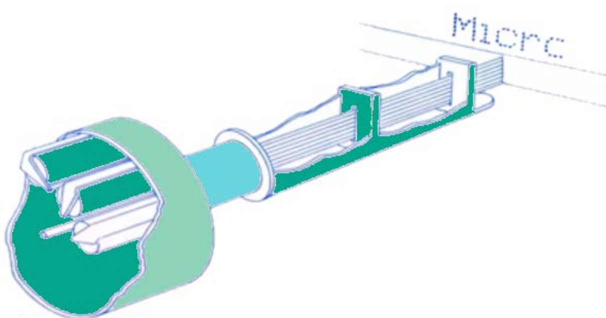


Figura 15.155 Impresora de matriz de puntos.

15.6.4 Inyección de Tinta

Una alternativa barata y de calidad media−alta es el uso de una cabeza que en lugar de contar con agujas que hagan impacto en una cinta de carbón, lance chorros de tinta directamente contra el papel a imprimir. La calidad de impresión va desde regular hasta buena−muy buena y se obtienen velocidades medias a altas con un precio medio.

Una de sus grandes ventajas es que el mecanismo de impresión no hace casi ruido y se puede usar papel estándar con formatos muy grandes sin necesidad de formas continuas. Esta ventaja es también su gran desventaja pues el uso de papeles de anchos no estándares no es posible más que en impresoras especiales y costosas.

15.6.5 Láser

Una innovación que tuvo su origen en las fotocopiadoras es la impresión por rayo láser. Debido al gran desarrollo del láser y de su constante disminución de precio y tamaño, la tecnología de las impresoras láser dejó de ser un artículo usado por las compañías de grandes recursos para convertirse, en menos de 1 año, en la elección para rapidez y calidad "casi de imprenta" de impresión.

La impresión por rayo láser consiste en al menos 6 fases:

- Cargar positivamente el rodillo de contacto.
- Remover la carga de las zonas de no impresión (las áreas blancas del papel) usando el láser.
- Revelar la imagen resultante usando carbón cargado negativamente.
- Transferir la imagen del rodillo al papel.
- Calentar el papel para lograr que el carbón quede firmemente adherido a la superficie del papel.
- Limpiar el tambor para prepararlo para la siguiente impresión.

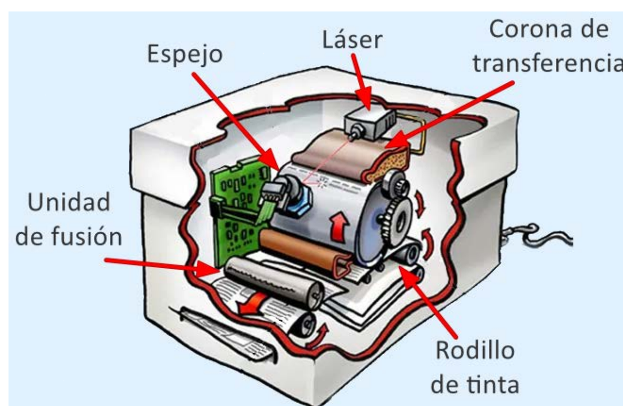


Figura 15.156 Impresora de rayo láser.

Las velocidades de impresión van de 4 páginas por minuto (ppm) a 20 ppm y su calidad, de 75 puntos por pulgada (ppi) a 3000ppi o calidad de imprenta.

Se espera que la tecnología láser sea substituida por la de rayo de electrones (de funcionamiento similar pero menos complejo) en un futuro próximo.

15.6.6 Graficadores

Para ciertos trabajos no es posible usar la impresora como tal y se prefiere otro tipo de dispositivos. Tal es el caso de dibujos o gráficas en los que la calidad debe ser excepcional o se debe usar un medio no manejado por las impresoras convencionales, como sería el caso de tinta china sobre papel especial.

Para el manejo de estos casos se usan los graficadores que consisten en plumillas de uno o varios colores que se mueven en dos planos controlados por un microprocesador que se comunica constantemente con el procesador central.

Dentro de los graficadores tenemos los de cama plana y los de rodillo. Los de cama plana son una mesa grande (más de 1.20mts) o chica (tamaño carta o A4) de trabajo donde se coloca el papel y sobre el que se mueve un mecanismo de impresión formado por una barra longitudinal que contiene la plumilla y que tiene desplazamiento en un eje junto con otro mecanismo que permite a la plumilla desplazarse en otro eje a lo largo de la barra que la sostiene. Debido al gran tamaño requerido para trabajos grandes, los graficadores de cama plana son limitados a tamaños de hojas pequeños.

El graficador de ***rodillo*** se forma de una barra que contiene a una plumilla que se desplaza en un eje longitudinal y de un rodillo donde se coloca el papel a imprimir y que gira en ambas direcciones dando el movimiento en el otro eje.

En ambos casos se pueden usar medios de impresión muy variados, así como plumillas de varios tipos. Algunos de estos graficadores tienen un dispositivo que contiene plumillas de varios colores que pueden ser controladas por el mismo graficador.

La calidad de impresión es alta, aunque la velocidad es relativamente baja. El costo es bastante elevado por lo que sólo son utilizados en despachos de arquitectos o en un lugar donde se aproveche todo su potencial.

15.7 Otros Dispositivos

Así como se cuenta con dispositivos para la salida y el almacenaje de datos, es conveniente utilizar otros medios de entrada alternativos al teclado que nos facilite el trabajo de capturar datos o sencillamente el de escoger u organizar opciones que el procesamiento de información nos presenta. Así han surgido a lo largo de los años distintos dispositivos que cubren una u otra área en la que el tradicional teclado es lento o ineficiente.

Entre los dispositivos más utilizados hoy en día se encuentran el ratón, el digitalizador de imágenes (scanner), las tabletas de dibujo y las lectoras de código de barras.

Describimos a continuación brevemente cada uno de los dispositivos externos de entrada más usados.

15.7.1 Joystick

Como se analizará en el capítulo *16*, algunos tipos de transductores (que convierten de un tipo de unidad a otra) utilizan resistencias variables llamadas potenciómetros para convertir posición espacial en corriente eléctrica que pueda ser interpretada por la computadora. Si acoplamos una palanca mecánica a dos potenciómetros rotatorios, cada uno de ellos en forma perpendicular, podemos conocer la posición ***X–Y*** (hacia adelante o atrás, hacia la derecha o izquierda) de la palanca. Los joysticks son comúnmente usados en equipo gráfico y en juegos. La conversión (transducción) se logra acoplando la palanca de forma que gire el potenciómetro ***X*** con el movimiento a lo largo del eje ***X*** y al potenciómetro ***Y*** con su movimiento a lo largo del eje *Y*. La posición ***(x,y)*** de la manija del joystick o palanca de juego se conoce entonces por la salida de voltaje proporcional ***(v(x), v(y))*** de los dos potenciómetros. En la parte superior de la palanca se coloca normalmente uno o varios botones para distintas funciones (generalmente realiza la misma función que la tecla de retorno (enter, return) de un teclado). En el caso de juegos, los botones tienen distintas funciones que se configuran según se requiera.

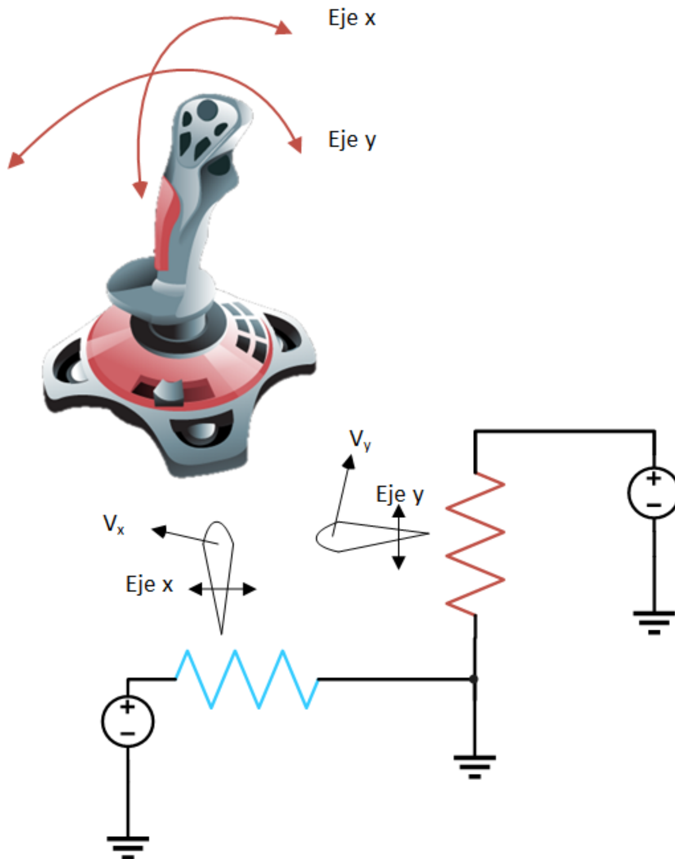


Figura 15.157 Mecanismo de un joystick.

15.7.2 Ratón

El ratón (mouse) es otro dispositivo señalizador que entrega una señal proporcional a su posición espacial ***X-Y***. Su funcionamiento es idéntico al de un joystick (sección anterior) pero en lugar de contar con una palanca acoplada a dos potenciómetros. Consta de una bola que hace girar en dos ejes unos discos con perforaciones que permiten o no el paso de la luz de un diodo emisor de luz (***LED***) infrarrojo hacia un fototransistor y su lógica que interpretan el movimiento del ratón. Todo el mecanismo se encuentra contenido en una pequeña caja de plástico que cabe en la palma de la mano cómodamente. En la parte superior del ratón hay normalmente dos o tres botones que realizan distintas funciones. Ciertos modelos incorporan una rueda que permite desplazarse rápidamente a través de un documento u otra función que el usuario define.

La entrada a la computadora se realiza por medio del puerto serial (lo más común) o por una tarjeta especial que realiza la conversión analógica-digital (ver capítulo *16*).

Existe otro tipo de ratón casi en desuso que usa una alfombrilla (placa, lámina) con un fino enrejado que al mover el ratón refleja o no la luz de su mecanismo óptico. Tiene la desventaja de que sólo es posible utilizar el ratón sobre esa rejilla especial y que el mecanismo es mucho más caro que el de su contraparte mecánica.

Existen también ratones inalámbricos que se comunican a la computadora por medio de señales fotoeléctricas infrarrojas que son interpretadas por un receptor conectado al puerto serial[62] o ***USB*** de la computadora.

Otros modelos no cuentan con piezas mecánicas y usan láser o cámaras para detectar cambios de la superficie sobre la que se mueven.



Figura 15.158 Ratón y su alfombrilla.

15.7.3 Digitalizadores

Es común el necesitar imágenes (y últimamente voz, música, animación) junto con el texto o información a procesar en un sistema informático. Cuando no se cuenta con el medio adecuado, es necesario dejar el espacio y posteriormente pegar la imagen, pero esto generalmente no da resultados convincentes. Un digitalizador de imágenes o scanner convierte tonos de blanco y negro a

[62] En desuso.

señales eléctricas por medio de un transistor fotoeléctrico. Estas señales son interpretadas por una tarjeta que las convierte de una cantidad analógica a digital (ver capítulo *16*), que luego son usadas por un programa especial en la computadora para armar la imagen a partir de la información entregada por el digitalizador.

Un proceso adicional consistente en usar otro programa de aplicación llamado reconocedor óptico de caracteres (***OCR***), permite convertir la imagen de texto y números en texto y/o números para su uso posterior en otras aplicaciones.

Los modelos más caros constan de filtros rojo, verde y azul, así como de tres juegos de fototransistores que permiten interpretar estos tres colores por separado y poder entonces digitalizar una imagen de color.

Existen principalmente dos modelos:

1. Digitalizadores de mano. Son de tamaño y precio reducido, pero su calidad deja mucho que desear pues son movidos manualmente sobre la superficie.
2. Digitalizadores de cama plana. Similares a una fotocopiadora; su precio es alto pero la calidad que se obtiene es de resolución fotográfica. Usualmente se emplea el mecanismo (en los modelos más caros) de estos aparatos, que es similar al de una fotocopiadora, como impresoras, fotocopiadoras, fotocomponedoras y digitalizadores a la vez; todo controlado por la computadora con los programas adecuados que proporciona el fabricante.

15.7.4 Otros (Guantes, palancas, plumas de luz)

Se han diseñado una multitud de métodos alternativos de posicionamiento o entrada de datos para aplicaciones específicas y especialmente para juegos. Entre los que destacan más se encuentran:

- Lectores ópticos de código de barras. Por medio de un fototransistor, convierte las barras negras y blancas a electricidad que, luego de ser convertidas en señales digitales, son interpretadas por un programa de aplicación especialmente diseñado. Entre los códigos de barras más usados están el EAN y el 5 de 2.
- Guantes. Un guante adaptado a la mano que conste de sensores de presión (ver capítulo 16) a lo largo del eje de

los dedos, puede convertir información espacial tridimensional en señales digitales fácilmente interpretadas por un programa de aplicación especial. Su uso va desde los juegos con realidad virtual (manipulación de objetos ficticios dentro de la computadora) hasta interpretación de lenguaje de señas para personas hipoacúsicas.

- Palancas. Una palanca que se gira o mueve con la mano obteniendo la misma información que la de un joystick, pero en tres ejes.
- Tableta de dibujo. Llamada también tableta digitalizadora, convierte la información introducida por medio de una pluma especial sobre una superficie plana, llamada tableta, a señales digitales usualmente introducidas por el puerto serial. Esta información es usada por programas de dibujo facilitando enormemente la introducción de datos gráficos.
- Pluma de luz. Pluma que consta de un fototransistor en la punta que al aplicarse contra la pantalla de video nos informa qué tanto tarda el haz de electrones en llegar de la parte superior de la pantalla (inicio de barrido vertical) a ese punto. Esta información nos da, de una forma indirecta, la posición de la pluma en la pantalla. Se usaba anteriormente para programas gráficos o como señalizador de opciones de programas, actualmente ha sido substituida por el ratón y la tableta de dibujo.



Figura 15.159 Guante, pluma de luz y otros.

15.8 Resumen

Se presenta en este capítulo toda una serie de dispositivos que, aunque popularmente se consideran parte de la computadora, son en realidad externos al sistema y pueden o no estar presentes sin que por eso se afecte el desempeño o funcionamiento de un sistema de cómputo.

Los dispositivos externos hacen más fácil el uso de la computadora y es casi impensable un sistema de cómputo que no cuente con por lo menos uno de los dispositivos aquí mencionados. Al ir bajando los precios y evolucionando las computadoras, los programas que se usan, los dispositivos que se consideraban como opciones por ser muy sofisticados o caros, se incorporan ahora como parte del diseño inicial y forman un paquete que sin duda nos hace más agradable usar las computadoras.

15.8.1 Puntos Importantes del Capítulo

- La interfaz forma el lazo de unión entre el mundo externo y la computadora.

- Las fuentes de poder forman un dispositivo externo no opcional pues sin ellas no funcionaría el sistema. Hoy en día se usan casi exclusivamente las fuentes reguladas por interrupción.
- El reloj, como la fuente de poder, el monitor y el teclado, son externos al sistema, pero imprescindibles. El teclado se forma por una serie de interruptores que se cierran al presionar las teclas a las que se unen mecánicamente; la información que generan es interpretada de forma programática o electrónica.
- Los monitores pueden ser tan sencillos como una serie de ***LEDs*** que nos indiquen el estado de varias funciones o tan complejos como un monitor ***LCD*** o ***LED*** que nos den muchísima información de una sola mirada. Las pantallas táctiles se utilizan en muchos dispositivos portátiles.
- Los dispositivos de almacenaje son necesarios en todo trabajo serio y permiten guardar la información tanto generada por el sistema de cómputo como la necesaria para que funcione; como los programas y sistemas operativos complejos. Pueden ir desde un disco duro de mediana capacidad hasta un sofisticado sistema de almacenaje por medio de discos de estado sólido externos y discos duros mixtos.
- Casi todo sistema grande tiene como salida gran cantidad de información que estamos acostumbrados a recibir de forma impresa. Las impresoras toman información generada por el sistema de cómputo y la traspasan a papel utilizando técnicas que van desde jet de tinta impulsado sobre el papel; hasta un rayo láser que fija una imagen positiva en un papel para luego ser revelada con carbón.
- Existe otra serie de dispositivos que hacen más fácil la tarea de introducir información de varios tipos a la computadora. Uno de los más comunes hoy en día es el ratón, pero se deben considerar otros muchos como joysticks, digitalizadores, guantes, palancas, plumas de luz, etc.

16 Otros Elementos Lógicos y Electrónicos

16.1 Interruptores Digitales y Analógicos

Las formas de onda digitales, por lo menos idealmente, realizan transiciones abruptas entre dos rangos fijos de voltajes. Un rango representa el valor 1 mientras que el otro representa el nivel lógico de 0. Dentro de cada rango, el valor exacto del nivel de señal no es significativo. En las compuertas lógicas, todas las entradas y salidas son señales digitales.

Los voltajes analógicos, por otro lado, son voltajes en los que su valor preciso es siempre significativo. Tales voltajes analógicos pueden ser de un valor fijo o pueden variar a través de un rango continuo de valores. Existe frecuentemente la necesidad de interruptores en circuitos y sistemas que usan voltajes analógicos y que deben ser controlados por una cantidad digital. Los circuitos de este tipo son llamados compuertas analógicas, compuertas de transmisión, compuertas lineales, circuitos de selección de tiempo, etc. dependiendo del propósito para el cual se usa el circuito. A la señal de control digital se le conoce como señal de compuerta, señal de control o entrada lógica.

En la figura *16.1* mostramos varios tipos de interruptores de este tipo en un esquema funcional. Una señal de control que realiza la transición entre dos estados digitales controla la apertura o cierre del interruptor. El mecanismo exacto de control depende del dispositivo utilizado para realizar el interruptor. Para este propósito se utilizan, entre otros, los interruptores digitales ***SCR*** y los ***Triacs*** (Silicon Controlled Rectifier; Thyristor Bidirectional Triode) o rectificadores (diodos) controlados por silicio que pueden funcionar transportando corriente en uno (***SCR***) o en ambos sentidos (***Triac***) una vez activados, al implicárseles una pequeña corriente de control. También es común usar los relevadores mecánicos (analógicos). Se prefiere evitar estos últimos precisamente por ser mecánicos y basados en un pequeño transformador y consumir, proporcionalmente, mucho más corriente.



Joseph Henry
(1846-1878)

Físico estadunidense que descubrió la autoinducción y el principio de inducción electromagnética. La unidad de medida de inducción eléctrica se denomina Henry (Henrios) en su honor. Experimentó y mejoró el electroimán, inventado en 1823 por el inglés William Sturgeon. Ya él había desarrollado electroimanes con gran poder de elevación. En 1831 creó el primer telégrafo electromagnético operativo. Henry también diseñó y construyó uno de los primeros motores eléctricos.

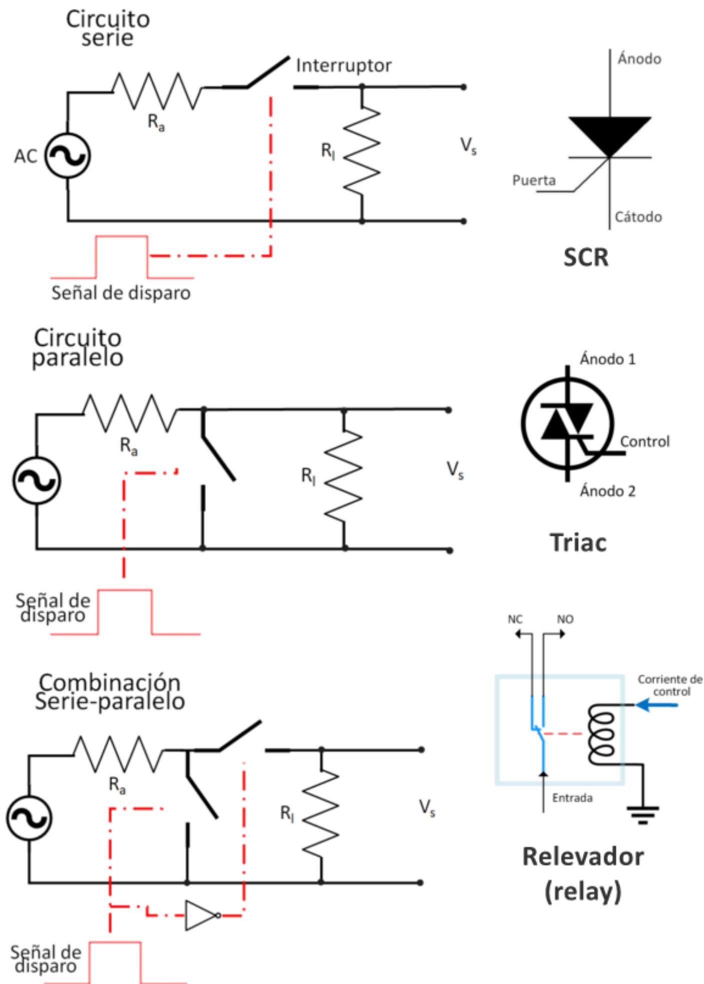


Figura 16.160 Interruptores digitales y analógicos.

16.2 Multiplexor

Una aplicación directa de los interruptores analógicos es su uso como mecanismo para conectar varias señales analógicas, una a la vez, a una carga común o hacia la entrada de otro circuito. A esta operación se le nombra ***multiplexar*** la señal. Cuatro señales de voltaje separadas se indican en la figura *16.2*, pero el número de señales puede variar de dos hasta varios miles. Para el circuito mostrado en la figura *16.2*, un contador de anillo de cuatro estados puede ser utilizado para controlar al interruptor.

La multiplexión se divide en:

- Multiplexión en tiempo. Cada señal toma su turno en un tiempo finito que depende del diseño, la aplicación y el número de señales que se requiera controlar.
- Multiplexión en frecuencia. Cada señal se transmite a la vez, pero cada una de ellas usa una frecuencia separada de las demás por un proceso de montar la señal sobre otra llamado modulación en frecuencia (FM).

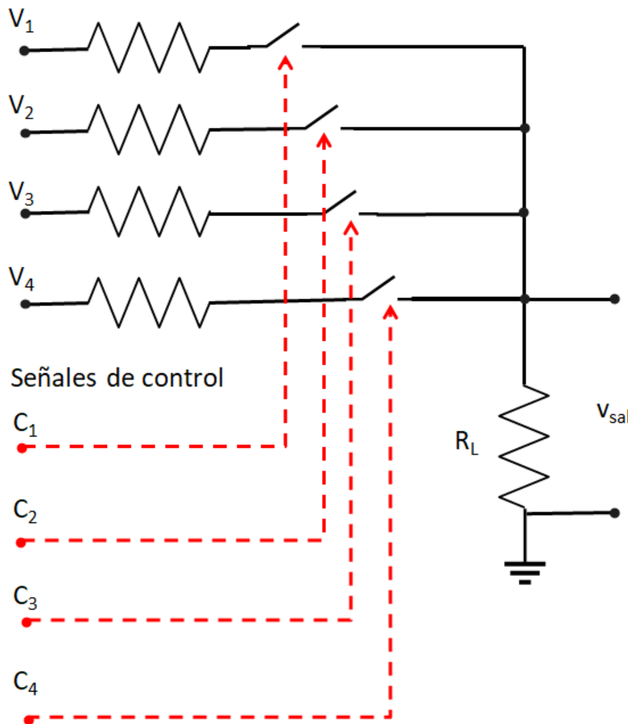


Figura 16.161 Multiplexor.

La utilidad de multiplexar una señal puede entenderse en los siguientes ejemplos:

1. Suponga que se tienen 5 llamadas en un sistema telefónico en el que sólo se cuenta con un medio para su transmisión. Una solución sería no permitir que se conectara más de un usuario a la vez, pero esto resultaría costoso y poco práctico[63]. Una mejor solución consiste en muestrear cada señal (de acuerdo con ciertas reglas conocidas como teorema de Nyquist) y luego enviarlas, una a la vez, por el mismo canal cada una ocupando un tiempo finito.

[63] Este era el caso en las primeras centrales telefónicas.

Del lado receptor debemos saber el orden de las señales y la frecuencia con que se realiza el muestreo y la transmisión para poder reconstruir la señal. Hemos resuelto el problema usando la multiplexión en tiempo.

2. Tenemos 5 estaciones de radio que quieren transmitir por un mismo medio (el aire) al mismo tiempo. Como en el caso anterior, una solución sería tomar turnos, pero nuevamente, esto no es práctico. Si encontráramos la forma de montar estas señales sobre otras que no interfirieran unas con otras podríamos resolver el problema usando la multiplexión en frecuencia.

16.3 Muestra y Retiene

Una segunda aplicación común de los interruptores analógicos consiste en muestrear una señal y retener su valor un tiempo finito mientras que el circuito que requiere esta señal pueda atender sus otras funciones y luego regresar a interpretar la señal. Está claro que requerimos un circuito que pueda "recordar" la señal para que ésta no se pierda si su duración es menor que este tiempo de atención finito.

Una aplicación directa de los circuitos de muestra y retiene es en la conversión de señales analógicas a digitales (ver siguiente sección). Otra aplicación frecuente es en la modulación codificada en pulsos (*PCM*). Aquí la forma de onda retenida se convierte de analógica a su equivalente digital. Para una discusión más a fondo vea la bibliografía, en especial *Telecomunicaciones y Teleproceso*.

16.4 Conversión Analógica Digital y Digital Analógica

En un sistema de cómputo requerimos frecuentemente interactuar con variables externas que no son voltajes digitales, sino cantidades analógicas. Para poder trabajar con estas cantidades necesitaremos forzosamente traducirlas de alguna forma a números digitales que sí puedan ser interpretados por la computadora.

Una función básica de los dispositivos de entrada/salida que forman la interfaz, es traducir la información de forma que sea entendida por los distintos componentes de un sistema computacional.

Por ejemplo, un teclado realiza un mapa entre la posición física de los dedos del operario y los voltajes digitales que representan esas

posiciones. Cuando tratamos con cantidades que representan valores continuos en el tiempo como es el caso de:

- Posición en el espacio
- Velocidad
- Aceleración
- Temperatura
- Presión
- Tasa de flujo de un líquido
- Intensidad de radiación (luz, radiación infrarroja, etc.)

Los dispositivos de entrada analógicos se requieren usualmente cuando una variable física continua se debe medir con exactitud; mientras que los dispositivos de salida analógicos son usados cuando se quieren controlar variables continuas de cantidades físicas. El control y la medición analógica son encontrados comúnmente en casi todas las aplicaciones de computación y pueden ser tan complejos como el control de procesos o tan sencillos (relativamente) como el control de una impresora.

Los requerimientos generales de una interfaz hacia una computadora son mostrados en la figura *16.3a*. Una señal no eléctrica analógica continua es primero convertida a una señal eléctrica analógica equivalente por medio de un transductor o sensor. La salida de este transductor debe ser acondicionada, esto es amplificada o reducida y filtrada, para igualar las características eléctricas de la siguiente etapa y que ésta pueda aceptar la señal sin sufrir daño alguno. El siguiente paso es convertir la señal analógica a una digital proporcional y esto se realiza con un convertidor Analógico-Digital (convertidor *A/D*). La señal se encuentra ahora lista para ser alimentada al sistema de cómputo para su procesamiento.

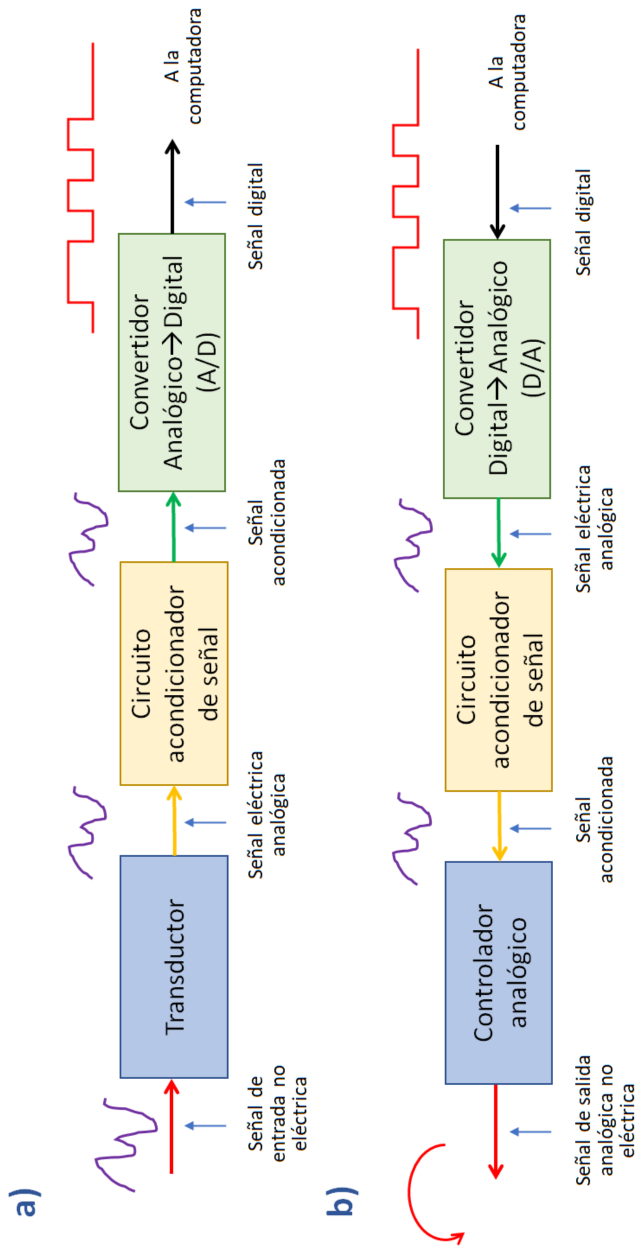


Figura 16.162 Interfaz analógica y digital.

El camino contrario se muestra en la figura *16.3b* donde el único cambio es el convertidor Digital-Analógico usado en lugar del *A/D* del proceso anterior.

Debe hacerse notar que, en muchos casos de interés, las salidas analógicas pueden ser controladas por señales digitales por lo que pueden ser tratadas en casi la misma forma por las interfaces digitales o puertos de E/S disponibles en el equipo. Un caso sencillo

se muestra en las figuras *16.4a* y *16.4b* donde en el primer caso la computadora muestra la temperatura en un despliegue digital y en el segundo forma un equipo de control de ***lazo cerrado*** de temperatura donde las variables son continuamente censadas y corregidas. Los distintos elementos que forman el sistema son explicados en las siguientes secciones.

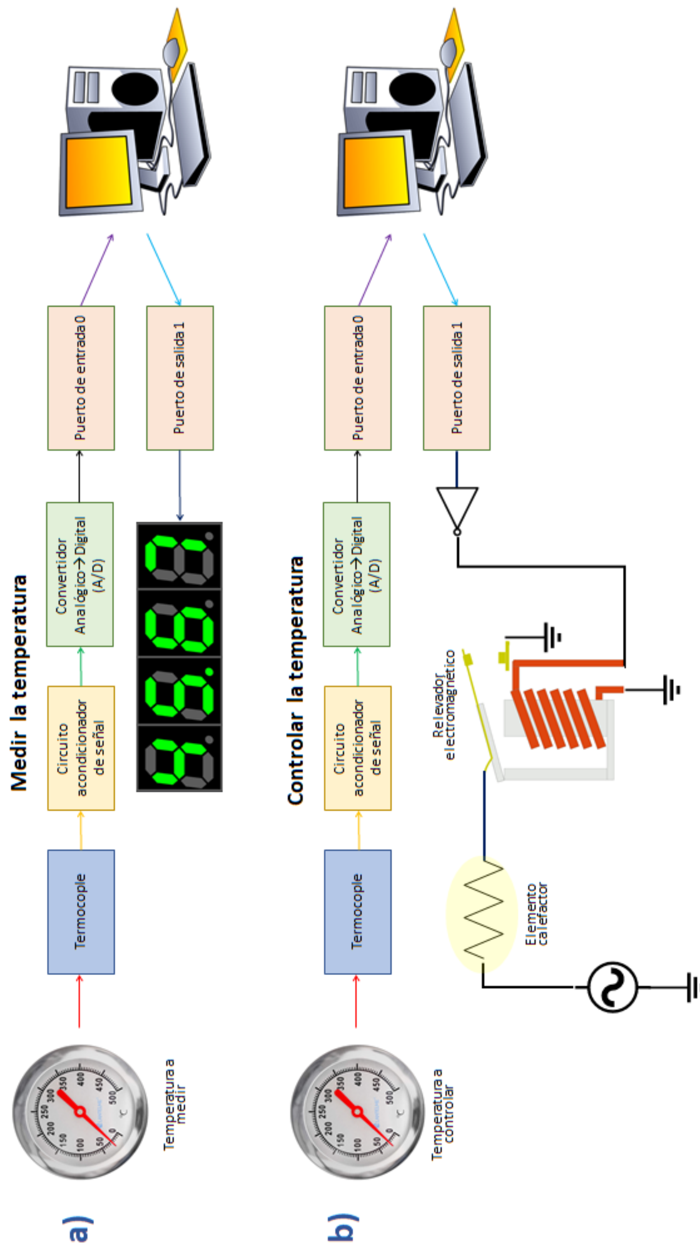


Figura 16.163 Despliegue y control de temperatura.

Problema

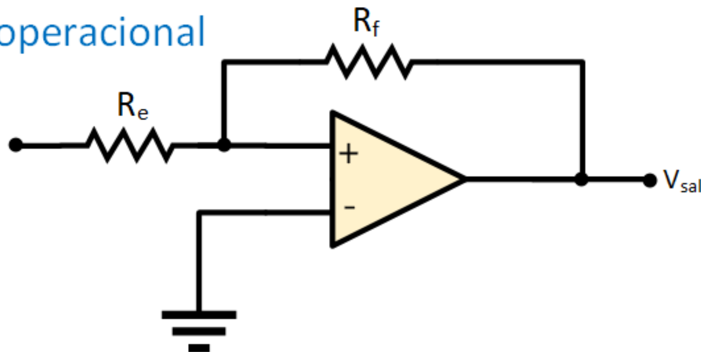
16.1 Proponga un programa de control para supervisar constantemente la temperatura de un horno y corregirla, como en el caso de la figura *16.4b*. Realice el programa en pseudocódigo.

Intente hacer una parte del código del programa en lenguaje de máquina de un circuito 80x86 ¿Qué tan difícil es esto?

Existen numerosas técnicas para convertir los datos de Entrada/Salida entre cantidades digitales y analógicas. Cada método tiene distintos compromisos entre velocidad y exactitud, así como también entre la complejidad de la electrónica utilizada o los programas que realizan la conversión. Aunque las conversiones analógicas/digitales (*A/D*) o digitales/analógicas (*D/A*) pueden realizarse con circuitos integrados específicos, también es posible realizarlo con circuitos de interfaz muy sencillos y un programa más o menos complejo que realice la conversión propiamente dicha.

La base de todos los circuitos usados es el ***amplificador operacional***, un circuito muy versátil que, entre sus funciones más importantes, está la de aceptar dos entradas de voltaje y entregar a su salida la suma o resta de éstas. El amplificador operacional[64] es usado también para realizar integrales y diferenciales con voltajes, siendo la base de muchas computadoras analógicas (casi en desuso) y filtros de señales usados en comunicaciones y electrónica analógica. Su amplificación puede llegar a factores de 100,000 (a relativamente bajas corrientes y voltajes) o más y esta característica lo hace ideal para servir de interfaz entre distintos tipos de sensores que tienen una salida de voltaje muy pequeña, que requiere amplificación antes de ser usada. Una aplicación de interés en electrónica digital es como comparador donde la salida del amplificador es 1 si el voltaje de entrada es mayor que un voltaje de referencia y 0 en el caso contrario.

[64] El más común de ellos es el IC741.



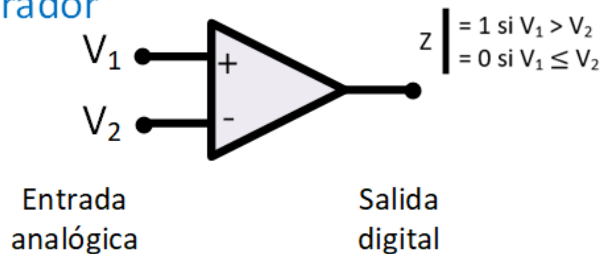


Figura 16.164 Amplificador y comparador.

Usando un comparador podemos realizar un sencillo circuito de conversión ***A/D*** como el de la figura *16.5* donde se tiene un simple modelo de alarma que funciona cuando la salida de voltaje del sensor sobrepasa un nivel establecido.

Un método de conversión ***D/A*** muy usado es el de la ***escalera de resistencias*** donde se usan resistencias que duplican su valor conforme se sube en la escalera como se muestra en la figura *16.6a*. La resistencia equivalente del circuito es:

(1.18) $$16.1 \quad \frac{1}{R_e} = \frac{x_{n-1}}{R} + \frac{x_{n-2}}{2R} + \cdots + \frac{x_0}{2^{n-1}R}$$

(1.19) $$\frac{x_0}{2^{n-1}R}(x_{n-1}2^{n-1} + x_{n-2}2^{n-2} + \cdots x_0)$$

por lo que

(1.20) $$16.2 \quad R_e = 2^{n-1}({}^{R}/_{N})$$

que implica que R_e es inversamente proporcional a la magnitud de ***N*** que es la palabra digital de entrada. La forma de conexión

para realizar la conversión completa puede verla en la figura *16.6b*.

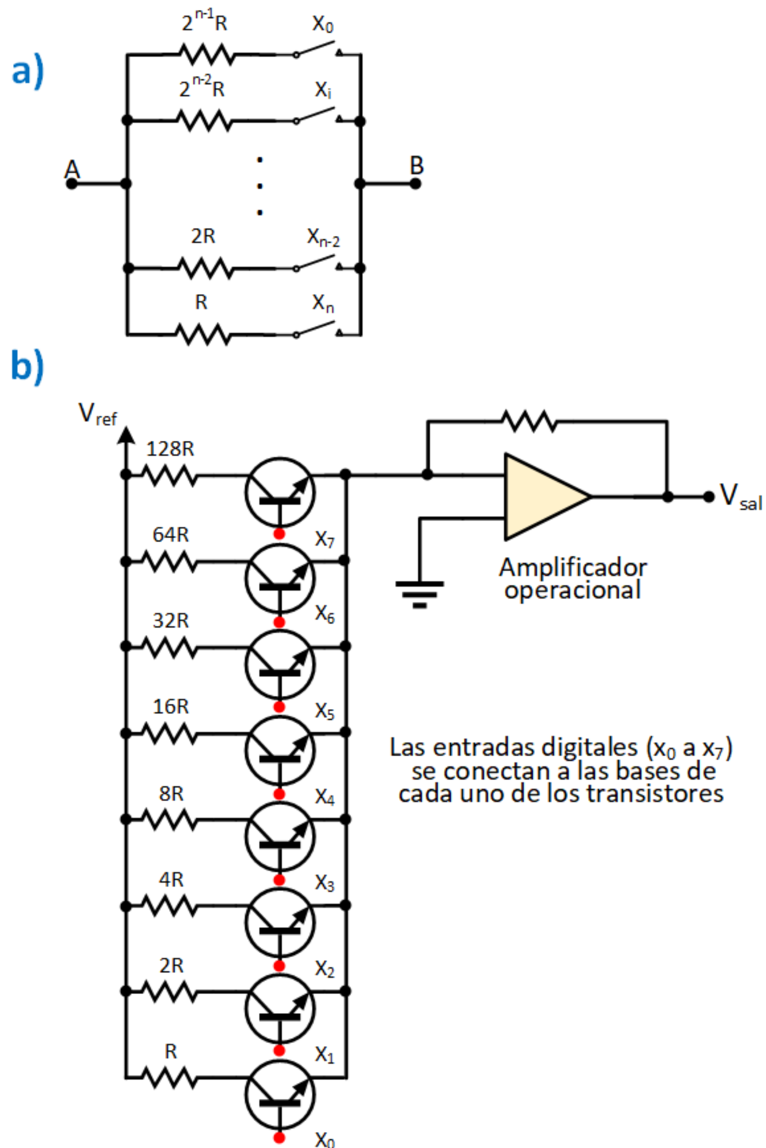


Figura 16.165 Conversión digital → analógica.

Una forma de conversión sencilla de ***A/D*** llamada ***directa***, se muestra en el esquema de la figura *16.7* donde el voltaje de entrada es comparado con voltajes de referencia que genera el propio circuito de conversión. Puesto que el número de comparadores crece exponencialmente con *n*, donde *n* es el número de bits o precisión requerida, la técnica se limita a 3 ó 4 bits haciendo que su precisión sea baja.

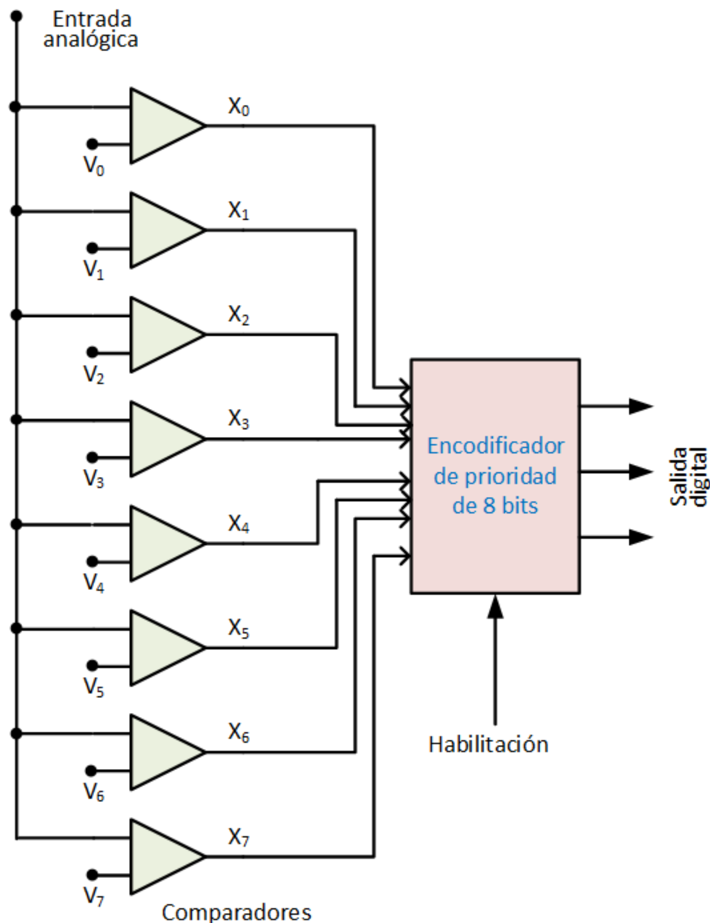


Figura 16.166 Conversión directa.

La conversión descrita en el párrafo anterior es muy rápida pues compara simultáneamente el voltaje desconocido con todos los voltajes digitales de referencia. Varios convertidores *A/D* usan un método indirecto donde el voltaje de entrada es comparado con uno de referencia V_{ref}. Este voltaje de referencia se modifica continuamente hasta que sea lo más próximo posible al voltaje de entrada. La conversión es más lenta pero la precisión obtenida es mucho mejor y la parte electrónica es más sencilla y barata. La conversión de esta forma se conoce algunas veces como ***conversión de rampa*** pues V_{ref} se incrementa en cantidades constantes y la forma de onda obtenida se asemeja a una rampa. El método usa tanto una parte electrónica como un programa de control.

Se conoce como ***resolución*** a la menor cantidad medible en un intervalo; la resolución aumenta en los convertidores al incrementar el número de bits de salida usados en la conversión. La ***precisión*** tiene que ver con la exactitud de la medición, por lo que un

instrumento puede ser preciso y de baja resolución o de alta resolución y poco preciso. Para una discusión más amplia vea la bibliografía sobre *Metrología* y la sección *1.6* (*Medición*).

Un algoritmo más rápido usa otro método conocido como de ***aproximaciones sucesivas*** en el que los bits de salida se obtienen uno a uno con la comparación del voltaje de referencia y el de entrada. Este procedimiento es más rápido, pues en lugar de realizar *2n* comparaciones como en el método de la rampa, se realizan sólo ***n***.

16.5 Transductores

Un transductor tiene la función de convertir de una variable de entrada no eléctrica a cantidades eléctricas, usualmente analógicas, que pueden ser leídas por un instrumento de medición o cualquier otro aparato que acepte entradas eléctricas. Existen una gran cantidad de métodos de transducción muchos de los cuales son extremadamente ingeniosos y sencillos. Muchos de ellos son tan simples como una resistencia variable a ciertas condiciones como a la temperatura (termistores) , la humedad, la tensión (válvula de esfuerzo) o la luz (fotorresistencia). Se muestran algunos de ellos en la figura *16.8*.

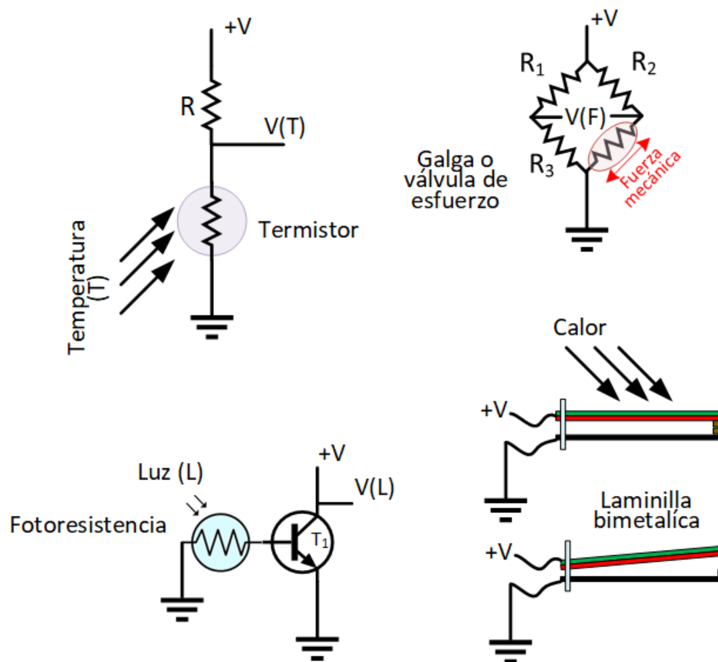


Figura 16.167 Distintos transductores.

Otros transductores usan resistencias variables o potenciómetros que son resistencias que constan de un contacto que, al deslizarse por su superficie, recoge la señal a distintas distancias variando la resistencia en forma proporcional. Este contacto se mueve usualmente por un brazo mecánico conectado físicamente a la variable a medir como el nivel del agua o la posición espacial de un objeto en dos o tres planos.

Una ventaja de usar la entrada como alimentación a un sistema de cómputo es que éste puede desentrañar las complejas variables del diseño mecánico pudiendo hacer muy sencillo el sistema de medición y dejando la complejidad al programa de interpretación.

Una aplicación de las fotorresistencias es su uso como medidores de velocidad en autos y bicicletas que puede traducirse a otras variables como aceleración, distancia recorrida, tiempo, promedios varios, etc. Su uso es muy efectivo y común pues elimina todo contacto físico entre las partes mecánicas en movimiento y la parte eléctrica que se mide (formando el sensor). Otra aplicación muy común es la lectura de códigos de barras para la venta o el control de distintos productos.

La temperatura se mide y convierte de muchas formas, pero las más comunes son usando el pirómetro, la cinta bimetálica, el

termistor y los termocoples. La cinta bimetálica forma un efectivo interruptor binario en el cual se permite o no el paso de corriente por un circuito formado de dos metales distintos, unidos firmemente o soldados. Cada uno de los metales tiene un coeficiente de dilatación térmica distinto. Una vez calibrado el dispositivo, la temperatura hace que una de las caras tienda a curvarse hacia la capa de metal que tiene menor coeficiente de dilatación, abriendo el circuito e impidiendo el paso de la corriente. Si los metales se enfrían vuelven una vez más a su posición original, cerrando el circuito nuevamente. Un uso común es el control de temperatura de planchas y cafeteras.

Ejercicio

16.1 Investigue distintos tipos de pirómetros, termistores y termocoples. De un ejemplo de su uso en circuitos comunes y sencillos.

16.6 Actuadores

El caso contrario a un transductor es cuando requerimos que una señal eléctrica sea convertida en movimiento mecánico para posicionar u orientar un objeto. Un ejemplo sencillo puede ser el de un relevador donde la señal eléctrica se traduce a movimiento mecánico que cierra o abre un interruptor (ver sección *2.3*). Los movimientos más complejos se controlan con motores eléctricos[65] y solenoides que nos entregan mucha más potencia para un trabajo mecánico.

Un solenoide es un tipo sencillo de actuador muy similar a un relevador pues se forma de un electroimán. Contiene un cilindro de alambre enrollado aislado que al hacerle pasar una corriente, funciona como un fuerte imán que atrae un pasador deslizante. El pasador regresa a su posición original al dejar de actuar el campo magnético del imán y por medio de un resorte que lo detiene firmemente (ver figura *16.9*).

[65] En serie, en paralelo o paso a paso.

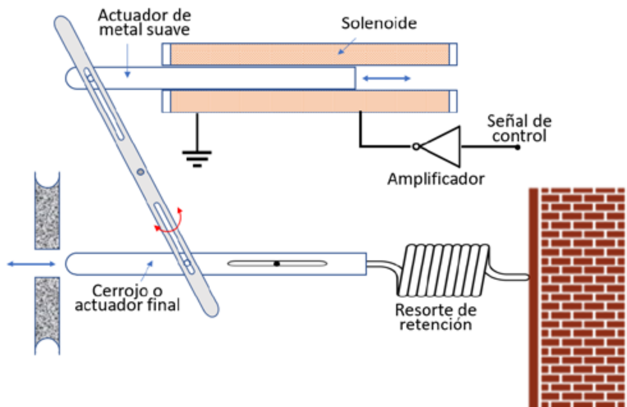


Figura 16.168 Solenoide.

El solenoide tiene muchísimas aplicaciones prácticas y de las más comunes son su uso para bloquear puertas eléctricas, disparar los alambres de impacto de una cabeza de impresión, mover el papel en casi cualquier tipo de impresión, etc.

Los motores eléctricos se mueven al variar sus campos magnéticos con respecto a su eje de rotación o rotor, aunque este principio puede variar grandemente con el tipo de motor que se trate: de jaula de ardilla, de imán permanente, de paso a paso, etc. Los actuadores se utilizan para dibujo mecánico, graficadores, herramientas de corte automatizadas, robótica, etc.

Ejercicio

16.2 Investigue el uso de motores de paso a paso (stepping motors) y su aplicación. Investigue la forma de interfaz con su contraparte digital.

16.7 Circuitos de tiempo

Como hemos evocado a lo largo de todo este libro, un circuito digital síncrono, que forma la mayoría de los sistemas computacionales actuales, depende enteramente de una base de tiempo. Las bases de tiempo deben tener varias características:

- Estabilidad. El circuito debe entregar la misma forma de onda y no depender de factores externos como pueden ser el voltaje entregado, la temperatura o la carga del circuito que genera la forma de onda del reloj.
- Confiabilidad. El circuito debe ser confiable bajo toda circunstancia. Esta característica se hace aún más crítica en

circuitos de tiempo que van en frecuencias mayores a 20 MHz, donde cualquier falla del reloj, por mínima que sea, causa un caos en la información que depende de esta base de tiempo.

- Ciclo de trabajo exacto. En muchos circuitos la forma de onda del reloj debe aproximarse en lo más posible a una onda cuadrada con 50% del tiempo en 1 lógico y el otro 50% en 0 lógico. Si los tiempos de 1 y 0 varían, se varía el ciclo de trabajo que es la relación entre el tiempo que un circuito de tiempo entrega su onda en 1 y el tiempo que esta onda está en cero para cada ciclo completo de reloj.

El circuito de tiempo puede ir desde un sencillo circuito formado por un capacitor y resistencia que oscila, hasta un complicado y exacto circuito formado por cristales de cuarzo y varios circuitos que aseguran su exactitud.

16.7.1 Multivibradores

Si recordamos el circuito flip-flop (llamado también multivibrador biestable), éste consta de dos estados estables, uno de ellos conocido como 1 lógico y el otro como 0. Un multivibrador monoestable es un circuito que consta de un estado estable permanente y otro semi estable. Para que se realice una transición de su estado estable al casi-estable se requiere de una señal externa. El circuito puede permanecer en el estado semi estable por un tiempo finito que es grande, comparado con el tiempo de la transición de un estado a otro. Eventualmente el circuito regresa a su estado estable sin requerir para ello de ninguna señal externa.

Como el circuito regresa a su estado estable por sí mismo después de un tiempo finito (controlable por diseño), se conoce a este circuito como mono pulso (one shot) puesto que genera una señal rectangular que puede ser usada como entrada de control para una siguiente etapa; por esto se le conoce también como circuito de compuerta (gating circuit). Más aún, como genera una rápida transición en un tiempo *T* posterior a la señal de disparo se le conoce también como circuito de retraso (delay circuit).

Sin la ayuda de una señal de disparo externa, una configuración astable realiza cambios sucesivos de un estado a otro, ***oscilando*** entre un estado y otro. Es precisamente este comportamiento el que nos interesa para los circuitos de tiempo.

Las compuertas del tipo *CMOS* son adaptables con facilidad para que tengan este comportamiento como se muestra en la figura *16.10*.

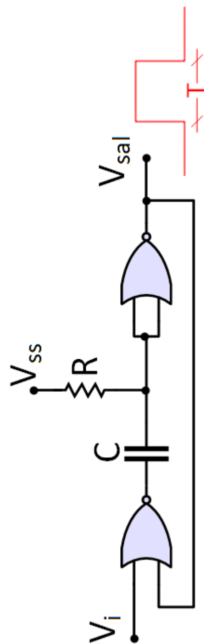


Figura 16.169 Multivibrador monoestable.

En esta figura tenemos dos compuertas del tipo ***NO O*** retroalimentadas con un juego de resistencias y capacitores. El uso de diodos limita el voltaje a los rangos que los circuitos pueden soportar sin ser destruidos. El tiempo de respuesta del estado semi estable depende de los valores seleccionados de *C* y *R*.

El mismo circuito conectado de distinta forma (figura *16.11*) nos entrega una señal oscilante cuadrada que puede usarse como base de tiempo.

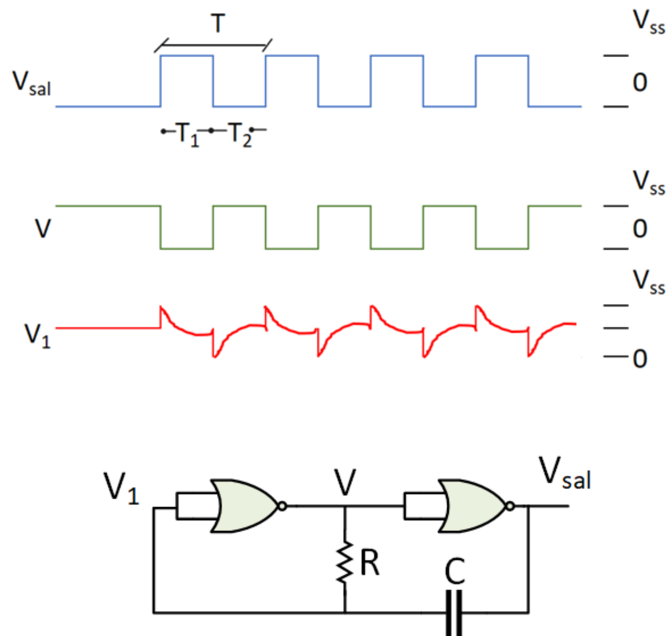


Figura 16.170 Multivibrador astable.

Para mejorar las características de estabilidad, ciclo de trabajo o confiabilidad se pueden realizar configuraciones alternativas con circuitos del tipo *ECL* o *TTL* como se muestra en la figura *16.12*. Se deja al lector interesado la búsqueda de estos temas en la bibliografía mencionada en el apéndice *E*.

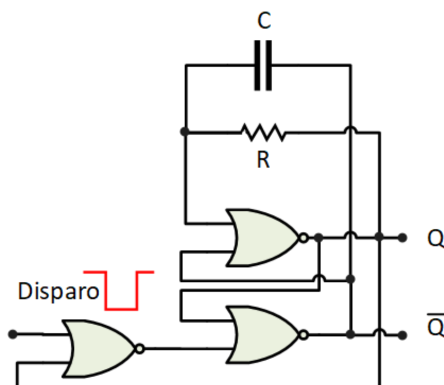
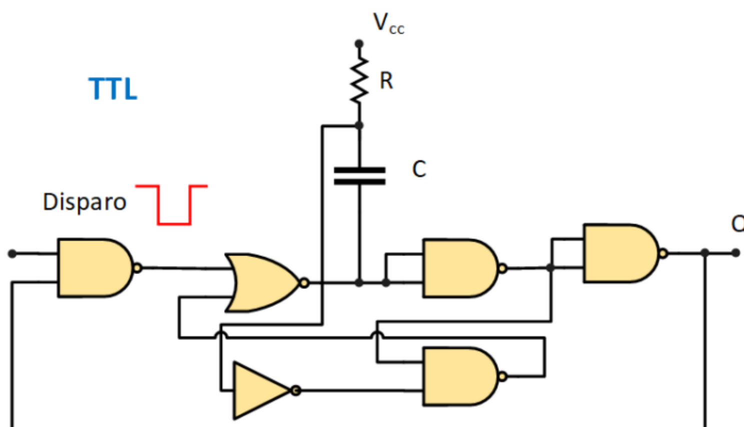


Figura 16.171 Multivibrador realizado con compuertas ECL y TTL.



El circuito *555* fue creado en 1970 por Hans R. Camenzind y comercializado en 1971 por Signetics (ahora NXP Semiconductors). Este circuito se sigue usando debido a su facilidad de uso, bajo costo y estabilidad. Se fabrican mil millones de unidades al año. El *555* contiene 23 transistores, 2 diodos y 16 resistencias que forman 4 elementos: dos amplificadores operacionales de tipo comparador; una puerta lógica de tipo inversor; y un flip-flop tipo *SR*. El *555* puede funcionar en tres modos: monoestable, astable o biestable.

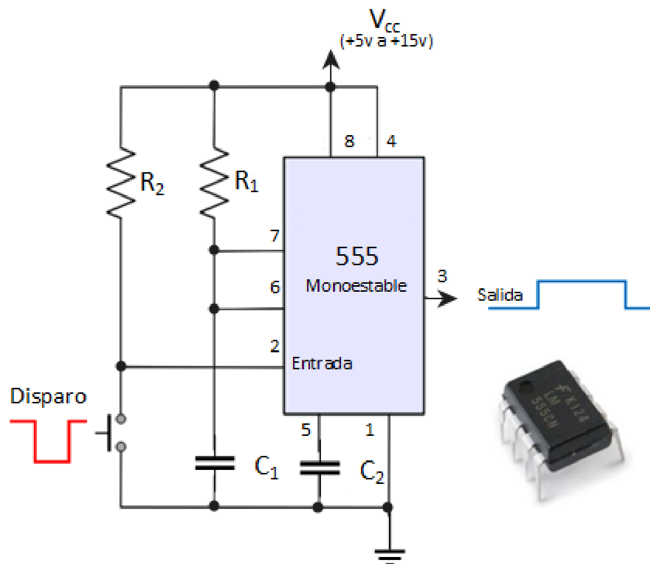
16.7.2 Circuitos Integrados de Tiempo

Es tan común el uso de circuitos de mono pulsos y osciladores, que pronto se diseñó un circuito de fácil aplicación que realiza ambas cosas según una sencilla configuración externa de resistencias y capacitores.

Uno de los principales y de mayor uso es el circuito *555* que funciona como monoestable y multivibrador astable según la configuración externa que se seleccione (vea la figura *16.13*).

El circuito es muy útil en distintas aplicaciones y su versatilidad lo ha hecho muy popular. Escogiendo las resistencias y capacitancias externas cuidadosamente podemos variar su funcionamiento en un amplio rango que va de un disparo cada pocos milisegundos hasta uno cada mes o año (limitado por la corriente de fuga del

capacitor seleccionado). El circuito también puede funcionar como un oscilador con frecuencias entre 0.1 Hz y 100 KHz.



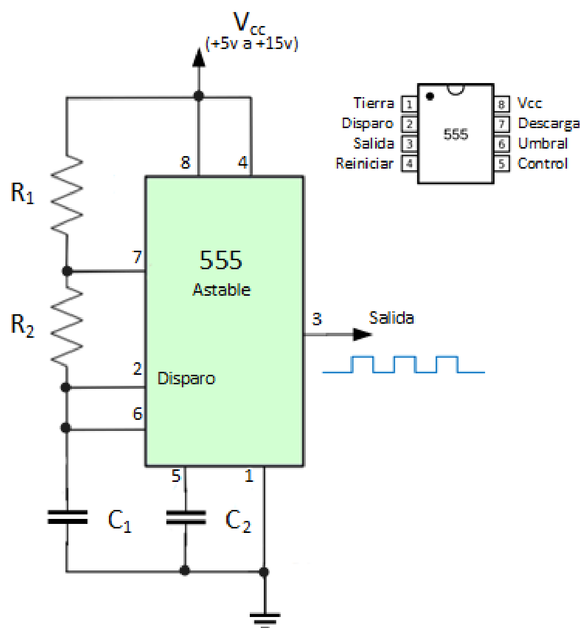


Figura 16.172 Circuito 555.

Problemas

16.2 Encuentre las ecuaciones de diseño para un circuito 555 en su forma monoestable, astable y biestable.

16.3 Con las ecuaciones encontradas en el problema anterior, encuentre los valores de R_1, R_2 y C_1 que nos den un reloj de 10 MHz con un ciclo de trabajo de 50%.

16.7.3 Cristales

La aceleración del reloj (overclocking) consiste en aumentar la frecuencia del reloj de un componente, ejecutándolo a una velocidad más alta de la que fue diseñado para funcionar. Esto generalmente se aplica a la *CPU* o *GPU*, pero también se puede usar en otros componentes.

Esta acción hace que el *CI* realice más operaciones por segundo, pero también genera más calor. Esta aceleración puede ayudar a obtener más rendimiento, pero a menudo necesitará enfriamiento adicional y más cuidado.

Cuando se requiere de gran precisión, estabilidad, bajo mantenimiento en ajustes y calibraciones y frecuencia de oscilación en una base de tiempo, no basta con usar un circuito tal como el 555 que depende de un oscilador tipo *RC*. Se procede entonces a usar un circuito cuya base de tiempo suele ser un oscilador piezoeléctrico de cuarzo. Los cristales de cuarzo poseen la propiedad de vibrar de forma muy precisa a altas frecuencias cuando se conectan a una fuente de voltaje en forma de sándwich (muy similar a la forma de un capacitor con una oblea de cristal de cuarzo sirviendo de dieléctrico). Esta característica los hace ideales para su uso como bases de tiempo.

Las frecuencias de oscilación van desde unos pocos kHz hasta varios cientos de MHz. Se acostumbra a dividir y acondicionar la frecuencia resultante para que los errores que puedan surgir del cristal se disminuyan considerablemente y se obtenga una onda cuadrada con un ciclo de trabajo del 50%. Los *BIOS* modernos permiten ajustar precisamente las velocidades del reloj del sistema, en especial la del *CPU* (overclocking), para sacar el máximo provecho; muchas veces con resultados desastrosos.

Los cristales de cuarzo son la forma estándar de generar señales oscilantes que sirvan de bases de tiempo. Mostramos un ejemplo de su interconexión en la figura *16.14*.

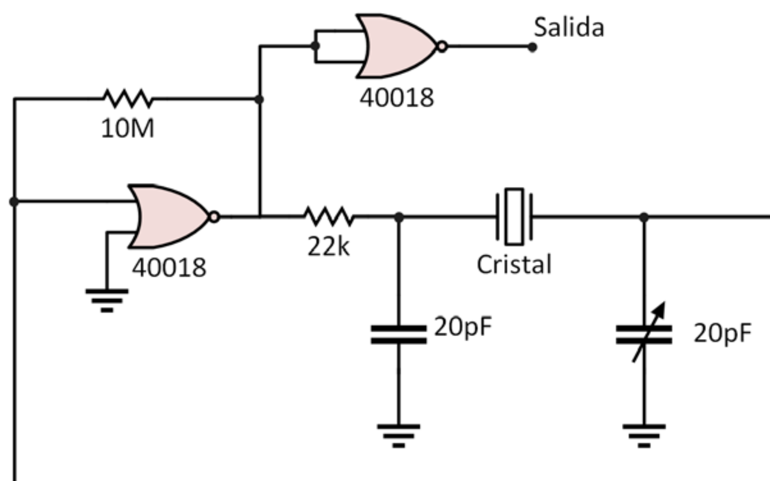


Figura 16.173 Oscilador con cristal de cuarzo.

16.8 Resumen

Se introducen en este capítulo otros circuitos útiles de los que ya se ha hecho mención a lo largo del libro. Los circuitos mencionados nos dan la pauta final para comunicarnos del sistema de cómputo al mundo externo y realizar otros tipos de aplicaciones que no se encuentren limitados al mundo virtual de la computadora.

16.8.1 Puntos Importantes del Capítulo

- Los voltajes digitales realizan transiciones bruscas entre dos estados nombrados arbitrariamente; mientras que los analógicos varían continuamente con el tiempo y pueden tomar cualquier estado de un conjunto infinito de posibles estados.
- Una aplicación directa de un interruptor analógico es el multiplexor donde varias señales se aplican a un sólo conjunto de cables que transmiten la información.
- Para poder hacer una interfaz entre el mundo real y el de la computadora, necesitamos convertir las unidades usadas por uno a las del otro. Los circuitos de conversión digital-analógico y viceversa nos dan la forma de realizarlo.
- Los transductores convierten de una unidad a otra, usualmente de cantidades físicas a eléctricas.
- Un actuador convierte energía eléctrica a mecánica para poder realizar un trabajo. Son utilizados en forma intensiva en los mecanismos de impresión.
- Los circuitos de tiempo forman el corazón de un sistema de cómputo como actualmente se conoce. Los más utilizados por su estabilidad, fiabilidad, exactitud y precio son los de base de cristal de cuarzo.
- Los circuitos de tiempo se configuran de dos formas: Monoestable donde una señal externa dispara un evento de tiempo finito después del cual el circuito vuelve a sus estado estable y Astable donde el circuito pasa de un estado a otro continuamente formando un tren de pulsos que no se detiene nunca.

A Glosario

Término en español	Término en inglés	Definición
Acarreo, Arrastre	Carryover	Mandato que indica que debe efectuarse un acarreo al siguiente dígito en una suma.
Acceso directo	Direct access	Característica de ciertas memorias en las que los datos almacenados en ellas pueden serlo en una secuencia de direccionamiento que no guarda criterio alguno y cuyo acceso se puede realizar en forma directa, reduciéndose así el tiempo invertido.
Acceso múltiple, Multiacceso	Multi-access	Sistema que permite a diversas personas realizar las funciones interactivas, generalmente relacionadas con la consola del operador.
Acoplador acústico*	Acoustical coupler	Dispositivo para interconectar el micrófono de un teléfono a una vía de acceso de entrada a una computadora (usualmente un módem).
Acoplamiento, Interconexión	Interfaz	Conexión común a dos sistemas distintos de computadores o a dos partes de un mismo sistema. Conceptos comunes que relacionan a dos equipos de diferentes funciones.
Actualizar	Update	Proceso de modificación de un archivo con otra información utilizada en un proceso.
Acumulador	Accumulator	Registro en el que se suma el resultado de una operación aritmética o lógica.
Adquisición de datos	Data acquisition	Obtención simultánea de datos desde dispositivos externos.
Algebra de Boole	Boolean algebra	Algebra capaz de poner en forma de ecuaciones las proposiciones lógicas, cuyos factores de operación son Y (AND), O (OR), NO (NOT), NO-O (NOR), etc. Desarrollada por Georges Boole.
Ajustar, Nivelar, Filtrar	Smooth	Acción de aplicar procedimientos para disminuir o eliminar las fluctuaciones rápidas en los datos.
Alfanumérico	Alphanumeric	Conjunto de todos los caracteres alfabéticos y numéricos.
Algoritmo	Algorithm	Conjunto de reglas escalonadas tendientes a la solución de un problema (que debe terminar en un tiempo finito).
Alineación de caracteres	Character alignment	Posición de los caracteres en relación al eje real de la línea de impresión.

Término en español	Término en inglés	Definición
Almacenamiento auxiliar	Auxiliary storage	Unidad de almacenamiento masivo externa distinta a la memoria principal.
Almacenamiento cíclico	Cyclic store	Forma de leer la memoria en el que el acceso a las posiciones de la misma se realiza periódicamente. Ejemplos clásicos de estas memorias son los tambores magnéticos.
Almacenamiento de acceso secuencial	Sequential access storage	Sistema de almacenaje en el que el acceso a los datos sólo puede efectuarse en la secuencia en que fueron guardados (tarjetas, cintas magnéticas, etc.).
Almacenamiento de gran capacidad	Bulk storage	Sistema de almacenaje masivo, cuyo tiempo de acceso es superior al de la memoria principal a la que complementa.
Almacenamiento de núcleo magnético*	Magnetic core storage	Conjunto de núcleos magnéticos dispuestos en matrices para formar la memoria de una computadora.
Almacenamiento en paralelo	Parallel storage	Sistema de memoria en el que el tiempo que se necesita para consultar una de sus zonas es el mismo que se requiere para acceder cualquier otra zona de la misma memoria.
Almacenamiento masivo	Mass storage	Sistema de almacenaje auxiliar de gran capacidad que está conectado directamente con una ***UPC***.
Almacenamiento principal	Main storage	Almacenaje desde el cual se ejecutan las instrucciones. Por lo general, es la memoria más rápida de una computadora.
Almacenamiento temporal	Temporary storage	Posiciones de memoria reservadas para los resultados intermedios.
Almacenamiento, Memoria	Storage	Dispositivo diseñado para aceptar la introducción y retención de datos para su posterior recuperación.
Ampliación	Add-on	Circuitería o sistema que puede adjuntarse a una computadora para ampliar memoria o prestaciones.
Amplificador de un solo extremo	Single-ended amplifier	Amplificador que desarrolla solamente una señal de salida.
Analizador	Analyser/analyzer	Cualquier dispositivo que controla un componente, tarjeta o sistema y muestre los datos analizados.
Analógico	Analog/analogue	Perteneciente a datos representados en forma de cantidades físicas continuamente variables que, si bien no se pueden contar, sí se pueden medir. Este concepto contrasta con el de digital. Por ejemplo, voltajes, corrientes, medidas angulares, etc.

Término en español	Término en inglés	Definición
Analógico-digital	A/D	Conversión de los voltajes y corrientes analógicas procedentes generalmente de un sensor para su representación digital en sistemas informáticos.
Archivo de cinta*	Tape file	Conjunto secuencial integrado por registros grabados en cinta.
Archivo de tarjetas*	Card file	Conjunto de tarjetas perforadas que guardan una información concreta, ordenadas con arreglo a una secuencia determinada a su utilización en una tarea específica.
Archivo, Fichero	File	Bloque lógico de información designado por un nombre y considerado como una unidad por el usuario. El archivo puede estar dividido físicamente en registros, bloques o en otras unidades.
Área de datos	Data area	Porción de memoria utilizada para contener los datos de cálculo durante la ejecución de un segmento.
Área de trabajo	Work area	Zona de memoria cuyo propósito es el almacenamiento temporal de los datos durante un proceso.
Asignación de memoria automática	Automatic storage allocation	Denominación que recibe la acción de asignar memoria a las variables de forma automática.
Asíncrono	Asynchronous	Evento o dispositivo que no es síncrono con la unidad de tiempo que rige a la unidad central de proceso (o de otros procesos).
Atenuación	Attenuation	Pérdida de amplitud de una señal.
Autónomo, Independiente	Stand-alone	Dispositivo o sistema que opera independientemente de otro dispositivo, sin su ayuda o sin estar conectado a él.
Banco de datos	Data bank	Conjunto de información que puede ser procesable en una computadora, independientemente del medio de almacenamiento.
Barra de acoplamiento de aplicación general	GPIB	Nombre utilizado por la norma de acoplamiento de barras (buses) IEEE-4881 975.
Barra de tipos	Type bar	Elemento de tipo lineal que contiene todos los símbolos imprimibles.
Barra, canal, conductor común, Bus	Bus	Conexiones por donde las señales van desde su origen a sus destinos. Cada Unidad de Procesamiento Central (***UPC***) crea, en el caso más general, tres barras (buses): dirección de memoria, datos y control.
Baudio	Baud	Tasa de transmisión por segundo. Unidad básica de transmisión de información por segundo, en el caso binario corresponde a bits por segundo.

Término en español	Término en inglés	Definición
Baudot*	Baudot	Código antiguo utilizado en teletipos de nivel 5 y en equipos de télex.
Biblioteca	Library	Colección de programas.
Bidireccional	Bi-directional	Flujo de datos que puede circular por un cable en cualquier sentido.
Biestable	Bistable	Dispositivo que siempre se encuentra en uno de los dos estados posibles permitidos en lógica binaria.
Bifurcación	Branch	Instrucción idéntica a Jump (salto) que origina una transferencia de control a otra secuencia de programa.
Bipolar	Bipolar	Tecnología utilizada en la fabricación de circuitos integrados. Emplea elementos de conmutación de transistor y está basada en la utilización de portadores mayoritarios para conmutar y amplificar.
Bit	Bit	Contracción de binary digit (dígito binario). Unidad de información que puede adoptar dos valores o estados distintos usualmente nombrados cero y uno, verdadero y falso.
Bit de parada	Stop bit	Bit que indica el final de una transmisión serie asíncrona.
Bit de paridad	Parity bit	Bit de verificación o su complemento que se añade a los bits de un bloque de información para controlar la transferencia de dicho bloque entre las unidades del equipo, pero que no tiene valor como información.
Bit de verificación	Check bit	Bit asociado a un registro de información con el propósito de comprobar la ausencia de error en el mismo.
Bit menos significativo	Least significant bit (LSB)	Dígito binario que tiene el menor peso (usualmente el de la primera posición de derecha a izquierda).
Bits de zona	Zone bits	Bits que representan la parte no numérica de un carácter. Bits distintos de los cuatro que se emplean para representar un dígito en un código de octeto.
Bits por pulgada	Bits per inch	Medida que define la densidad de registro en una pista en una cinta magnética. Se abrevia BPI.
Bloque	Block	Unidad física de información en un registro lógico. Normalmente se expresa en bytes.
Bloque variable	Variable block	Bloque cuyo tamaño no es fijo, sino que varía (dentro de ciertos límites) de acuerdo con las necesidades de los datos.
Bloqueo	Lockout	Proceso que automáticamente se aplica en un sistema, mediante el cual una o varias partes del mismo se inhiben del resto de las operaciones durante el tiempo en que éstas no deben participar en ellas.

Término en español	Término en inglés	Definición
Borrado	Blanking	Función de suprimir en una unidad de presentación visual (pantalla), el haz de electrones a fin de evitar luminiscencia o brillo.
Borrar	Erase	Expresa la idea de suprimir los datos almacenados sobre un soporte cualquiera devolviendo cada uno de los elementos a su estado nulo original, incluso, a otro diferente al existente en ese momento.
Borrar, Restaurar	Clear	Acción de suprimir los datos de una memoria o de alguna dirección de memoria, sustituyéndolos por otros predeterminados.
Bucle (lazo) restaurador	Self-resetting loop	Conjunto de instrucciones secuenciales formando bucle, de las que algunas de ellas tienen por misión sustituir los datos e instrucciones modificadas a su valor inicial, cada vez que se empiece el bucle.
Bucle abierto	Open loop	Sistema de control, en el que la acción correctora no es automática, sino que depende de una intervención externa determinada por la información que aparece representada.
Bucle, Lazo	Loop	Circuito físico por el que es factible realizar una transmisión. Conjunto de operaciones que se realizan un número predeterminado de veces hasta cumplir con una condición.
Búsqueda	Fetch	Traer de la memoria hacia el ***UPC*** un conjunto de información de caracteres.
Búsqueda	Search	Investigación que se hace sobre un conjunto de elementos de información para seleccionar aquel o aquellos que reúnen las condiciones exigidas al ordenar la búsqueda. El elemento investigado puede estar almacenado en cualquier tipo de memoria.
Búsqueda binaria	Binary search	Técnica en la que el intervalo de búsqueda se divide por dos en cada iteración.
Búsqueda, Posicionar	Seek	Búsqueda física en una memoria de acceso aleatorio, situando en posición el mecanismo de acceso con anterioridad a la búsqueda lógica.
Cabezal*	Head	Dispositivo que lee, registra o borra datos en un sistema de almacenamiento magnético.
Caché	Cache memory	Memoria intermedia de alta velocidad, utilizada entre el procesador central y la memoria principal para acelerar la transferencia. Puede existir en varios niveles.
Cadena	Chain	Secuencia cíclica de bits que forman las palabras de un código encadenado. Dispositivo móvil empleado en las impresoras,

Término en español	Término en inglés	Definición
		que dispone, sobre unos eslabones colocados horizontalmente, de todos los caracteres existentes en el conjunto de impresión.
Cadena	String	Cualquier conjunto de elementos o unidades que han sido ordenados en una secuencia, según un orden específico. Cualquier conjunto de caracteres o dígitos consecutivos que se encuentran en el almacenamiento.
Cadena en margarita	Daisy chain	Mecanismo para priorizar interrupciones.
Cambio de escala	Scale	Proceso consistente en alterar las unidades en que se expresan las variables, con objeto de situarlas al alcance de la capacidad de la máquina o al programa de que se trate.
Canal	Channel	Conexión lógica entre una ***UPC*** y un dispositivo de E/S.
Canal acústico	Voice grade channel	Canal apto para realizar transmisión de voz.
Canal analógico	Channel Analog	Canal en el que la información transmitida es de naturaleza analógica.
Canal de cuatro hilos	Four wire channel	Circuito de doble vía sobre el que se transmiten señales simultáneamente por caminos separados y distintos, en direcciones opuestas, dentro del mismo medio de transmisión.
Canal de dos hilos, Par	Two wire channel	Circuito formado por dos vías de transmisión, la cual puede efectuarse en cualquier dirección, pero no simultáneamente.
Canal dúplex	Channel duplex	Canal sobre el que se puede realizar una transmisión simultánea en ambas direcciones.
Canal semidúplex	Half duplex channel	Canal sobre el cual se puede transmitir y recibir señales, pero solamente en una dirección a la vez.
Carácter	Character	Cada uno de los símbolos convencionales adoptados para representar, aisladamente o relacionados entre sí, la información. Representación efectiva o codificada de un dígito, letra o carácter especial.
Carácter de sincronización	Synchronization character	Carácter que se inserta automáticamente en la corriente de datos de un equipo de comunicaciones síncrono para mantener y establecer el sincronismo.
Carácter más significativo	Most-significant-character (MSC)	En la representación posicional, dícese del carácter situado en la posición extremo izquierda de un grupo de caracteres significativos (el de más peso).
Carácter numérico	Numeric character	Cualquier carácter empleado como dígito en la representación de números.

Término en español	Término en inglés	Definición
Caracteres por segundo	CPS	Tasa de transferencia de datos estimada a partir de la tasa de bits y la longitud de caracteres.
Carga de entrada	Fan-in	Carga eléctrica presentada por la salida de todo el conjunto de circuitos conectados a una entrada.
Carga de memoria	Memory fill	Técnica empleada para depuración de programas, mediante la cual se llena la memoria libre de unos caracteres que impedirán la ejecución del proceso si éste intenta ejecutar instrucciones en dichas áreas.
Carga de salida	Fan-out	Carga eléctrica que puede excitar la entrada de un conjunto de circuitos. Normalmente se expresa como el número de entradas (circuitos independientes del mismo tipo) que pueden ser excitadas.
Cargador	Loader	Programa especial que toma la información en formato binario y la pone en memoria.
Cargador absoluto	Absolute loader	Programa destinado a cargar un programa en una dirección numérica específica.
Carro	Carriage	Dispositivo existente en ciertos equipos impresores destinado a soportar y a controlar el papel donde se imprime la información.
Casete*	Cassette	Funda plástica de pequeño tamaño que contiene y conserva cintas magnéticas en su interior para evitar contaminación con partículas extrañas y servir de soporte al medio.
Celda de almacenamiento	Storage cell	Unidad elemental de almacenaje.
Celda de datos*	Data cell	Tarjeta magnética de almacenaje de datos masivo de la IBM. Usada en los años sesenta.
Célula binaria	Binary cell	Celdilla de almacenamiento capaz de retener un dígito binario.
Centinela, Señalizador	Sentinel	Carácter empleado para indicar la presencia de una condición específica; por ejemplo, el final físico de una cinta magnética o el final de un registro de longitud variable en el almacenamiento.
Cerdíp	Cerdip	Contracción de la denominación en inglés de cápsula cerámica con conectores (patas) en disposición DIP (paquete doble en línea, Ceramic Dual Inline Package).
Cero, Nada	Zero	Número que denota magnitud cero. Condición de códigos que una computadora reconoce como cero.

Término en español	Término en inglés	Definición
Cerrojo	Latch	Dispositivo físico que captura la información y la retiene.
CI a la medida	Custom IC	Circuito integrado fabricado según especificaciones del usuario.
Ciclo de ejecución	Execute cycle	Tercer ciclo de los tres ciclos para ejecución de instrucción de programa; durante este tiempo se lleva a cabo la instrucción propiamente dicha.
Cinta*	Tape	Término genérico por el que se conoce a los soportes de información formados por una cinta magnética o de papel.
Cinta de papel perforado*	Paper tape	Cinta de papel de tipo apergaminado destinada a recibir unas perforaciones realizadas en posiciones específicas que representan la información.
Cinta reutilizable*	Scratch tape	Cinta magnética que contiene información que ha perdido vigencia y puede borrarse y utilizarse de nuevo para almacenar nuevos datos.
Circuito amplificador de impulsos, Limitador	Slicer	Circuito que amplifica de forma eficaz una parte de los impulsos de llegada comprendidos entre dos niveles de amplitud espaciados entre sí con mucha proximidad.
Circuito multiterminal	Multistation	Circuito que interconecta varias terminales situadas en distintas posiciones, haciendo llegar la información transmitida simultáneamente a todos ellos.
Clasificar, Ordenar	Sort	Acción de disponer las unidades o elementos de información en grupos, según las claves de identificación de cada uno. Los elementos se ordenan en secuencia si la disposición de las claves sigue algún orden predeterminado.
Cociente	Quotient	Resultado de dividir un operando, denominado dividendo, por otro llamado divisor, denominándose resto al excedente que se produce cuando el dividendo no es múltiplo del divisor.
Codificación numérica	Numeric coding	Dícese de cualquier sistema de codificación que únicamente utilice números.
Codificación rectilínea	Straight-line coding	Codificación que evita el empleo de bucles, mediante repetición de partes de la codificación, siempre que sea necesario.
Codificación relativa	Relative coding	Escritura de instrucciones de programa que se lleva a cabo utilizando las técnicas del direccionamiento relativo.
Codificación simbólica	Symbolic coding	Escritura de un programa en un lenguaje fuente.
Codificador-decodificador	Codec	Dispositivo o programa que comprime datos para posibilitar una transmisión rápida y una descompresión de los datos recibidos.

Término en español	Término en inglés	Definición
Codificar	Encode	Transformar la representación de una información aplicando un código, pero sin modificar el contenido de dicha información.
Código americano normalizado para el intercambio de información	ASCII	Código de caracteres utilizado por la mayoría de equipos distintos a los fabricados por IBM y Western Digital (que usan los propios).
Código corrector de errores	Error correcting code	Código generado para detectar errores y corregirlos.
Código de autoverificación	Self-checking code	Sinónimo de código de detección de errores.
Código de datos	Data code	Conjunto de normas por las que transmiten, reciben y procesan las señales que componen bloques de información.
Código de dos a cinco	Two-out-of-five code	Código binario en que cada dígito decimal se representa por dos bits 1 y tres bits 0.
Código de edición	Edit code	Número o letra que indica que la edición ha de efectuarse según regla preestablecida.
Código de instrucciones simbólicas de carácter general para principiantes	BASIC	Lenguaje de alto nivel inventado por Dartmouth con fines instructivos. Es fácil de aprender y de usar y es el que guarda mayor similitud con el FORTRAN.
Código de máquina	Machine code	Sistema de codificación adoptado en el diseño de una computadora para representar su juego de instrucciones.
Código de redundancia	Redundant code	Código que utiliza más elementos de señal que los estrictamente necesarios para representar la información esencial para el proceso o a la que se ha de transmitir.
Código EBCDIC	EBCDIC	Código de 8 bits empleado por IBM para codificar símbolos alfanuméricos. Esencialmente es análogo al ASCII, pero con una codificación distinta.
Código fuente	Source code	Programa escrito por el usuario, usualmente en código ASCII, y que se introduce en el sistema que lo interpreta.
Código Hamming	Hamming code	Código que puede corregir algunos errores de forma automática.
Código objeto	Object code	Salida de un compilador o de un ensamblador, que, por su propia naturaleza, es un código ejecutable de máquina o resulta apropiada para que el programa ligador la

Término en español	Término en inglés	Definición
		procese con el fin de producir un código ejecutable de máquina.
Código relativo	Relative code	Código de programa en el que las direcciones se especifican en relación con alguna dirección de base o código en el que se emplean direcciones simbólicas.
Código unitario	Unitary code	Código que consta de un sólo código. La cantidad que representa está determinada por el número de veces que el código se repite.
Cola de espera	Queue	Grupo de elementos de un sistema que esperan su tratamiento.
Cola de espera en canal	Waiting queue channel	Grupo de artículos que se han de tratar en un sistema que trabaja en tiempo real, los cuales permanecen a la espera de que el programa controlador de canal los tome en consideración.
Cola de salida	Output queue	Término colectivo con el que se designan las colas de control que describen conjuntos de datos de salida del sistema.
Comando de canal	Channel command	Instrucción que indica a un canal, a una unidad de control o a un dispositivo una operación.
Compaginado, Fusión	Merge	Proceso que se utiliza para formar una secuencia clasificada de registros, partiendo de dos o más secuencias previamente clasificadas.
Comparación con selección	Matching	Técnica que consiste en comparar las claves de dos registros para seleccionar artículos con destino a una fase particular del proceso o para rechazar los registros inválidos.
Compilador	Compiler	Programa de traducción que convierte instrucciones de alto nivel en un juego de instrucciones binarias (código objeto) para su ejecución.
Complemento a diez	Tens complement	Operación resultante de sustraer cada digito de un número a la base del sistema menos uno, añadiendo después 1 al último dígito significativo.
Complemento a dos	Two's complement	Método de expresar números binarios en donde el negativo de un número se genera complementando al número y añadiendo la unidad.
Complemento de la base	Radix complement	Número obtenido al restar cada dígito de una cantidad a la base menos uno, añadiendo después 1 al dígito menos significativo, efectuando el acarreo de números necesarios.
Comprobación de caracteres	Character check	Comprobación que verifica la observación de las reglas que determinan la formación de caracteres.

Término en español	Término en inglés	Definición
Comprobación de paridad impar	Odd-parity check	Verificación de paridad en el que se espera que el número de unos o ceros de un grupo de dígitos binarios sea impar; incluyendo en la cuenta el propio bit de paridad.
Comprobación de resultados de salida	Check out	Verificación de los resultados a la salida de una computadora, localizando errores y efectuando las oportunas correcciones.
Comprobación paridad par	Even parity check	Verificación encaminada a averiguar si el número de dígitos 1 (o 0) de un grupo de dígitos binarios es par o impar.
Compuerta NO-O	NOR-gate	Elemento lógico en el que la variable representada en la única señal binaria de salida es la no disyunción de las variables representadas en las señales binarias de entrada.
Compuerta tipo "O"	OR-gate	Elemento lógico en el que la variable representada en la única señal binaria de salida es la disyunción de las variables representadas en las dos o más señales binarias de entrada de que dispone el elemento. Es decir, la señal binaria de salida será verdadera cuando una o más señales binarias de entrada sean verdaderas.
Compuerta, puerta	Gate	Circuito provisto de una única señal de salida, cuyo valor depende del estado de las señales de entrada.
Computador de uso general	General purpose computer	Computadora concebida para resolver una amplia variedad de aplicaciones de sistemas de procesamiento de datos. Contrasta con las de uso específico que sólo desempeñan una función determinada.
Computador, Ordenador	Computer	Sistema de cálculo de propósito general que incorpora una ***UPC***, memoria, dispositivos de E/S, fuente de alimentación y un chasis de soporte.
Computadora anfitriona, Computadora primaria	Host computer	Computadora utilizada para preparar programas que han de utilizarse en otra computadora o en otro sistema de procesamiento de datos.
Computadora de reserva	Standby computer	Computadora utilizada en un sistema dual o dúplex que está a la espera para hacerse cargo de la carga de proceso en tiempo real cuando sea necesario.
Comunicación binaria síncrona	Binary synchronous communication	Tipo de transmisión en el que se hace uso de la transmisión síncrona en binario.

Término en español	Término en inglés	Definición
Comunicación de datos	Data communication	Proceso consistente en transmitir y recibir datos que comprende operaciones tales como codificación, decodificación, etc.
Conductor común, Línea principal	Highway	Véase bus.
Conexión posterior (entre tarjetas)	Backplane	Área física donde se agregan las tarjetas de un sistema. Normalmente contiene las ampliaciones del sistema tanto en forma de circuito impreso o cableado.
Configuración	Environment	Estado de todos los registros, ubicaciones de memoria y de otras condiciones operativas de un sistema.
Configuración objeto	Object configuration	Conjunto de equipos de una computadora encargada de realizar el programa objeto.
Conjunto, Juego	Set	Colección o conjunto de elementos que tienen alguna característica en común o entre las cuales existe relación.
Consola	Console	Terminal que dispone de la mayoría de las funciones de control en un sistema.
Consulta de tablas, Búsqueda de tablas	Table look-up	Método de investigación de tablas para localizar elementos relacionados con una clave determinada. Operación de la búsqueda de la función de una tabla en memoria que corresponde a un argumento determinado.
Consulta, Interrogación	Inquiry	Petición de información contenida en la memoria.
Contador de décadas	Decade counter	Divisor por diez.
Control de paridad par-impar	Odd-even check	Forma de verificación de paridad en la que se agrega un bit adicional a una palabra o carácter; este bit tiene el valor 1 ó 0, dependiendo de que el número de bits 1 (0) de la palabra o carácter sea par o impar.
Control de trabajos agrupados en secuencia	Sequential-stacked job control	Sistema de control que asegura que los trabajos se ejecutan en la secuencia en que se presentan al sistema.
Control secuencial	Sequential control	Método de funcionamiento de una computadora en que las instrucciones se almacenan en el mismo orden en que se ejecutan.
Conversión binaria a decimal	Binary to decimal conversion	Proceso de conversión de un número binario a su equivalente decimal.
Copia de seguridad	Backup copy	Copia que contiene una reproducción o duplicado del conjunto de datos y/o programas que se están utilizando, y que se guarda como reserva para casos de pérdida o destrucción del original.
Corrida de prueba,	Test run	Prueba que se realiza para comprobar que un programa determinado está

Término en español	Término en inglés	Definición
Pasada de prueba		funcionando correctamente, y en la que se emplean datos de prueba para generar resultados que se comparan con las respuestas que se espera obtener.
Corriente de salida	Output stream	Mensajes de diagnósticos y otros datos que el sistema o un programa de proceso emiten en dispositivos de salida especialmente activados para este fin.
Corte	Cutoff	Punto en el que la degradación que experimenta una señal, originada por la atenuación o la distorsión, hace que la señal no resulte utilizable.
Cristal	Crystal	Cristal de cuarzo cuya piezoelectricidad (oscilación al aplicar voltaje) proporciona una frecuencia exacta para aplicar a la temporización del reloj. Se abrevia Xtal.
Cristal	Xtal	Véase cristal.
Cuantificador	Quantizer	Dispositivo que convierte una cantidad analógica a su equivalente digital.
Cuanto	Quantum	Cantidad elemental indivisible, por la cual se puede variar una magnitud física determinada.
Cuenta, Recuento, Cinta	Tally	Lista impresa de cifras que produce una máquina sumadora.
Cursor	Cursor	Punto móvil luminoso, usualmente mostrado en forma de línea o subrayado, en la pantalla que indica el lugar de emplazamiento del carácter siguiente que se introduce.
Curva de aprendizaje	Learning curve	Mejoras en el proceso de fabricación respecto a la experiencia. Tiempo comprendido entre aprender algo y poder usarlo de forma funcional.
Datos en bruto	Raw data	Datos que no han sido procesados por ningún sistema.
Decibelío	Decibel	Unidad que expresa en forma logarítmica las relaciones de tensiones, corrientes o potencias.
Decimal codificado	Coded decimal	Tipo de notación en el que cada dígito decimal se identifica mediante un grupo de unos y ceros binarios.
Decimal codificado en binario	BCD	Representación de los números decimales en la que cada dígito se representa mediante 4 bits.
Decodificador	Decoder	Unidad lógica que decodifica dos o más bits en salidas mutuamente exclusivas.
Demodulación	Demodulation	Proceso mediante el cual, se recupera información de una onda portadora modulada. Es el proceso inverso al de modulación.
Demultiplexador	Demultiplexer	Circuito lógico que puede escoger una señal de una línea que contiene varias y

Término en español	Término en inglés	Definición
		dirigirla a una línea única de información digital.
Depuración, corrección, puesta a punto	Debugging	Proceso de exploración, localización y corrección de errores en la programación o en el funcionamiento de equipos.
Descargar a la memoria externa	Roll-out	Volcar al exterior de la computadora el contenido de la memoria principal para registrarlo en un sistema de almacenamiento auxiliar.
Desempaquetar, Desagrupar	Unpack	Recuperar datos originales de una posición de almacenamiento en que han sido condensados en unión de otros datos.
Desfase	Offset	Diferencia entre el valor o condición buscada y el que realmente se obtiene. Desplazamiento en memoria.
Desmontaje	Takedown	Operaciones efectuadas al término de un ciclo de funcionamiento para dejar preparado el equipo para la siguiente puesta a punto.
Desplazamiento cíclico	Cyclic shift	Desplazamiento en el que los datos que salgan por un extremo del registro de memoria vuelven a entrar por el otro, como en un bucle cerrado.
Desplazar	Shift	Operación consistente en trasladar o mover los elementos de una unidad de información hacia la izquierda o hacia la derecha.
Desplazamiento de salida	Shift out	Acción de mover la información, dentro de un registro general, hacia uno de los extremos, a fin de que, a medida que la información sale por ese extremo, los ceros vayan entrando por el extremo opuesto.
Diafonía	Crosstalk	Interferencia entre dos circuitos o señales.
Diagnóstico	Diagnostics	Juego de rutinas empleadas para diagnosticar los fallos de funcionamiento de un sistema.
Diagrama de flujo, Organigrama	Flowchart	Representación gráfica de un programa, en el que se refleja la secuencia general de las operaciones. Para ello se utiliza una serie de signos convencionales, generalmente geométricos, enlazados por líneas de interconexión.
Digital/Analógico	D/A	Conversión de representación digital utilizada en equipo digital, para excitar cualquier otro equipo que use señales analógicas (motores, altavoces, etc.). Con esto, las computadoras pueden relacionarse con el mundo real.
Digitalizador	Digitizer	Equipo convertidor encargado de transformar las señales analógicas en digitales.

Término en español	Término en inglés	Definición
Dígitos significativos	Significant digits	Dígitos o posiciones de dígitos de un número cuyos valores se conocen y tienen relación con la precisión del número.
Dirección	Address	Número que indica la posición de una palabra o dispositivo de entrada-salida en memoria.
Dirección de tres más uno	Three-plus-one address	Relativo a una instrucción que contiene tres direcciones de operando y una dirección de control.
Dirección generada	Generated address	Dirección de memoria numérica o simbólica, generada por una instrucción de un programa. También se le conoce por dirección sintética.
Dirección múltiple	Multiple-address	Término relativo a las instrucciones que especifican la dirección de más de una posición de memoria.
Dirección relativa	Relative address	Dirección especificada en una instrucción que no ha sido modificada.
Dirección virtual	Virtual address	En los sistemas dotados de memoria virtual, dícese de la dirección que hace referencia a la memoria virtual y ha de convertirse a dirección de la memoria real como condición previa a su utilización.
Direccionamiento directo	Direct addressing	Técnica de direccionamiento empleada para instrucciones con saltos cortos, usualmente se incluye la dirección en una palabra siguiente (8, 16, 32 bits o más).
Disco flexible*	Floppy disk	Dispositivo de almacenamiento masivo que utiliza un disco flexible para registrar la información.
Disco magnético	Magnetic disk	Dispositivo de almacenamiento que consta de varios platos metálicos circulares planos que están revestidos en ambas caras por una capa de material magnetizable.
Diskette, Disco flexible*	Diskette	Disco de plástico mylar flexible recubierto de óxido magnético.
Disponibilidad de datos	DAV	Productos y servicios que aseguran que los datos informáticos continúen estando disponibles en cierto nivel de desempeño en situaciones que van de normales a aquellas de “desastre”.
Dispositivo autónomo	Off-line unit	Dispositivo E/S o equipo auxiliar que no está bajo control directo de la ***UPC***.
Dispositivo de llamada automática	Automatic calling unit	Dispositivo de marcación mediante el cual, un equipo de transmisión de datos puede generar automáticamente una llamada.
División o reparto del tiempo	Time-slicing	Sinónimo de tiempo compartido.

Término en español	Término en inglés	Definición
Documentación de programa	Write up	Conjunto de documentos necesarios para la utilización de un programa.
Dúplex	Duplex	Método de comunicación bidireccional que permite la transferencia simultánea de datos en ambas direcciones.
Dúplex total, Bidireccional simultáneo	Full duplex circuit	Dícese del circuito capaz de admitir la transmisión de señales en ambos sentidos y de forma simultánea
Duración de respuesta	Response duration	Intervalo que abarca el origen de tiempo de un impulso y el momento en que el impulso cae por debajo de un valor operativo específico.
Eco	Echo	Señal de conversación o de datos que se devuelve desde un punto distante con amplitud y retardo suficientes para molestar al que habla o interferir la transmisión normal de datos.
Editor	Editor	Programa designado para introducción de texto en un sistema informático.
EIA-RS232C	EIA-RS232C	Acoplamiento serie estándar para comunicaciones asíncronas.
Ejecución de un proceso, Pasada de máquina	Run	Utilización de una computadora para la obtención de un trabajo determinado. Pasada de un grupo de tarjetas perforadas por una máquina. Tratamiento de varias rutinas encadenadas automáticamente, durante el cual no es necesaria la interrupción del mismo por parte del operador.
Ejecutar un paso	Step	Acción de hacer que una computadora ejecute una operación.
Electrónica, Componentes, Dispositivos físicos	Hardware	Equipo físico contrapuesto a los programas de la computadora para su empleo.
Elemento umbral	Threshold element	Dispositivo que recibe una serie de señales digitales de valor 1 o 0 afectadas cada una de ellas de un peso específico diferente, de las que obtiene otra señal binaria de salida, su estado depende de la suma de los pesos específicos indicados y del valor 1 o 0 de dichas señales de entrada.
Empujar, Emparejar tarjetas	Joggle	Operación consistente en agitar con la mano un archivo de tarjetas perforadas para introducirlas alineadas en el almacén de alimentación de la máquina que ha de trabajar con ellas. Proceso de colocar parejas de tarjetas con arreglo a un criterio determinado.
Emulación	Emulation	Simulación en tiempo real. Un dispositivo A emula a otro tipo B mediante la

Término en español	Término en inglés	Definición
		ejecución de un emulador de forma que se comporta como el dispositivo B.
Emulación en circuito	In-circuit emulation	Instalación hardware-software para el depurado de las Entradas/Salidas en tiempo real. La ***UPC*** emulada puede detenerse y sus registros examinarse; los dispositivos de E/S pueden controlarse desde la consola del sistema de desarrollo y los programas pueden residir en memoria ***RAM*** (simulada) o en ***ROM***/***PROM***.
En línea, en conexión directa	On-line	Se dice que una parte de un sistema de computación está en línea, cuando se halla bajo el control directo de la ***UPC***.
Encaminamiento, Encauzamiento	Routing	Asignación de una vía de comunicaciones por la que un mensaje o llamada telefónica alcanzará su destino.
Enlace	Link	Indicador de dirección al próximo elemento de una lista o de un bloque contiguo de un archivo.
Enlace	Linkage	En programación, dícese del código que conecta dos rutinas codificadas por separado y que pasa los valores y/o el control entre ellas.
Enmascaramiento, Mascarilla	Masking	Técnica que consiste en utilizar una máscara para operar sobre la configuración de bits de algún otro operando, para alterar o aislar ciertas posiciones de bit.
Ensamblador	Assembler	Programa que toma la forma mnemónica del lenguaje simbólico y lo convierte en código objeto binario para su ejecución.
Entrada	Entry	Cada una de las instrucciones que generan una función dentro de un programa o de una subrutina. La primera instrucción de una subrutina es, forzosamente, una entrada, pudiendo existir dentro de la misma varias entradas más.
Entrada/Salida, E/S	I/O, Input/output	Líneas o dispositivos empleados para obtener o enviar la información al exterior de lo que propiamente se considera la computadora (***UPC*** y memoria).
Entrehierro	Gap	Espacio comprendido entre el cabezal lector o grabador y el soporte de registro magnético.
Entrelazar, Concatenar	Interlace	Operación consistente en unir dos archivos siguiendo una regla. En los monitores, cuando las líneas de barrido no son continuas, esto es, se generan primero las impares y luego las pares.
Error	Bug	Término general que indica la presencia de un error en un programa o el funcionamiento defectuoso del equipo.

Término en español	Término en inglés	Definición
Error	Error	Cualquier discrepancia entre los valores correctos y los obtenidos en algún tipo de cálculo o proceso.
Error de máquina	Machine error	Error introducido en los resultados de un proceso automático que puede atribuirse al funcionamiento defectuoso de una máquina, más que a un fallo en el manejo del equipo.
Error relativo	Relative error	Relación de un error existente en algún resultado calculando respecto al valor cuantitativo del resultado.
Error residual	Residual error	Error generado durante un experimento; es la diferencia entre un resultado exacto calculado teóricamente y uno obtenido empíricamente.
Error, Equivocación	Mistake	Fallo, cualquiera que sea la circunstancia, imputable a la responsabilidad humana.
Escala	Scale	Gama de valores aceptados determinada, frecuentemente, por la longitud de palabra de la computadora o por la rutina de que se trate.
Escape de enlace de datos	Data link escape	Carácter de escape utilizado para introducir la información de control de flujo de datos.
Escribir, Grabar	Write	Acción de transcribir datos en una forma de almacenamiento desde otra de almacenamiento distinta; por ejemplo, transcribir datos a una cinta magnética desde la memoria principal de una computadora.
Espera	Latency	En dispositivos de almacenamiento rotativos, dícese del retardo transcurrido entre el instante en que el dispositivo es notificado de la llegada de una transferencia y el instante en que el dispositivo está listo para realizarla.
Espera, En espera	Standby	Condición del equipo que permite reanudar por completo el funcionamiento uniforme y regular en un breve plazo de tiempo.
Estación sin operador	Unattended station	Estación repetidora que normalmente no está dotada de personal encargado de su atención y mantenimiento.
Estado	Status	Condición actual de un dispositivo.
Estilo	Style	En el reconocimiento de caracteres ópticos, dícese de las proporciones que distinguen a los caracteres y que permanecen constantes, cualquiera que sea el tamaño del carácter.
Explorar	Scan	Examinar cada uno de los elementos de una lista o los registros de un archivo, generalmente como parte de un sistema de recuperación de información en el que se comprueba cada unidad de información

Término en español	Término en inglés	Definición
		en orden para saber si satisface o no, determinadas condiciones. Acción de verificar el estado o condición de los enlaces de comunicación, o los canales de E/S a fin de determinar si los canales se están utilizando o no. Digitalizar una imagen para su procesamiento en la computadora.
Extractor	Extractor	Palabra situada en un registro que indica la parte de otra palabra con la que ha de llevarse a cabo la operación.
Fase de ejecución	Run phase	Término relativo a la compilación de programas. Se emplea para indicar el período en que el programa objeto compilado se prueba y procesa por primera vez.
Fase objeto	Target phase	Durante la compilación, fase en la que se procesa primeramente el programa objeto.
Fila	Row	Cada una de las líneas perpendiculares al eje longitudinal de una cinta de papel o magnética, sobre la que se pueden configurar las perforaciones o magnetizaciones de puntos correspondientes a la representación codificada de un carácter. Disposición horizontal de caracteres u otras expresiones.
Final	End	Instrucción (sentencia) que denota el final de un programa.
Flujo, Corriente	Stream	Datos que se están transfiriendo a/desde un medio de almacenamiento externo, representado como una serie de partidas de datos en forma de caracteres y prescindiendo de los límites de la línea de impresión o del tamaño del registro de entrada.
Formato de instrucción sin dirección	Zero address instruction format	Instrucción en cuya constitución no existe la parte de direccionamiento.
Formato de la etiqueta	Tag format	Disposición y escritura de un registro empleado como etiqueta para localizar una posición de excedentes.
Fragmentación	Fragmentation	Situación en la que una memoria masiva dispone de muchos espacios libres y necesita ser compactada. Algunas veces se usa una rutina especial llamada Recolector de basura (Garbage collection).
Fuera de línea, Autónomo	Off-line	Se dice que una parte de un sistema de la computadora está fuera de línea cuando no se halla bajo el control directo de la ***UPC***.
Función de transferencia	Transfer function	Expresión matemática que especifica la relación entre dos fenómenos que existen en puntos diferentes, en el tiempo o en el espacio, en un sistema determinado.

Término en español	Término en inglés	Definición
Funcionamiento sin operador	Unattended operation	Tipo de operación que, por los dispositivos automáticos de que está dotada una estación terminal, permite realizar la transmisión y recepción de datos sin la intervención humana.
Ganancia	Gain	Aumento del nivel de señal que se obtiene a través de un amplificador.
Generador	Generator	Rutina de control que efectúa una función de generación.
Generador de caracteres	Character generator	Circuito que forma las letras o números en una pantalla o impresora.
Generador de función natural	Natural function generator	Generador en el que la función se basa o en una formula o en una ley física o matemática.
Grabación externa	Outboard recorder	Rutina o módulo que registra los datos pertinentes en el archivo de registro del sistema cuando se produce un error de E/S no recuperable.
Grabación sin retorno a cero	Non-return-to-zero recording	Método de registrar magnéticamente bits, en el que la magnetización empleada para representar ceros y unos ocupa totalmente el elemento de almacenamiento y en el que no es posible volver a una condición de referencia entre bits.
Gráfica, Diagrama	Chart	Representación gráfica de un conjunto de datos.
Graficador, Trazador de gráficos	Plotter	Dispositivo que delinea o traza una imagen visual de una función o tabla.
Graficador, Trazador de gráficos X-Y	X-Y plotter	Dispositivo que se utiliza en combinación con una computadora para trazar puntos de coordenadas en forma de gráfico.
Habilitación, Desinhibición	Enable	Restauración al estado operativo de un dispositivo deshabilitado (inhibido).
Hexadecimal	Hex, Hexadecimal	Representación de los números en base 16.
Identificar	Identify	Asignar un código exclusivo a una información.
Identificarse	Log-on	Acto de presentación o autoidentificación por el que un usuario se hace reconocer por el sistema.
Impresión de salida	Printout	Término general con el que se designa la salida de una impresora; páginas impresas producidas por una impresora.
Impresora de impacto	Impact printer	Dícese de cualquier dispositivo mecánico de impresión en el que los caracteres se forman golpeando una cinta contra el papel.
Impresora de líneas	Line printer	Impresora de alta velocidad capaz de imprimir de forma simultánea una línea completa.

Término en español	Término en inglés	Definición
Impresora de páginas completas	Page-at-a-time printer	Tipo de impresora en el que la impresión puede efectuarse por páginas completas, si bien el formato de los caracteres de dicha página ha de ser definido con anterioridad.
Impresora xerográfica	Xero-graphic printer	Dispositivo de impresión que utiliza xerográficas y en el que se determina el formato de la impresión a realizar antes de ejecutarla.
Impulso de escritura	Write pulse	Impulso excitador que establece la condición 1 de una celda magnética o graba en ella.
Impulsor de cinta	Tape drive	Dispositivo perteneciente a una unidad de cintas magnéticas, cuya finalidad es mover las cintas bajo las cabezas de lectura-escritura.
Incrementar	Increment	Modificar una cantidad, sumándole otra.
Indexado	Indexed	Modo de direccionamiento en el que la dirección actual se obtiene por adición de un desplazamiento a la dirección de base.
Indicador	Pointer	Elemento de programa que hace referencia a otro.
Índice de utilidad	Serviceability	Fiabilidad del equipo, basada en algún criterio objetivo. Se adoptan diferentes criterios para determinar o valorar el índice de utilidad.
Indirecto	Indirect	Modo de direccionamiento en el que la dirección de la ubicación en memoria a ser leída o escrita, está contenida en otro lugar de memoria o en un registro.
Información inválida	Garbage	Término que expresa la idea de información sin sentido y, por tanto, no válida.
Informe	Report	Término genérico aplicado a cualquier análisis de datos impreso producido por una computadora.
Inicialización	Initialization	Poner en funcionamiento una computadora en un estado conocido.
Inmediato	Immediate	Modo de direccionamiento en que la dirección de la localidad de memoria a ser leída está contenida en la instrucción misma.
Instituto de Ingenieros Electrónicos	IEEE	Organización no lucrativa fundada en 1963 que trabaja para innovar, educar y estandarizar la industria del desarrollo eléctrico y electrónico.
Instrucción	Instruction	Sentencia única en un programa. Esta sentencia puede ser extraída de memoria, decodificada y ejecutada por la ***UPC***. Las instrucciones pueden ser aritméticas, lógicas, de operación con registros, memoria y dispositivos de E/S u operaciones de control específicas.

Término en español	Término en inglés	Definición
Instrucción de dos direcciones	Two-address instruction	Formato de una instrucción en cuyo contenido se incluyen dos direccionamientos.
Instrucción de longitud variable	Variable-length instruction	Característica que da por resultado un mayor aprovechamiento de la memoria principal, al utilizarse únicamente las posiciones de memoria exigidas por las instrucciones de la aplicación de que se trate.
Instrucción de máquina	Machine instruction	Instrucción escrita en código de máquina de una computadora; es decir, aquella que la máquina puede realizar directamente sin necesidad de traducción.
Instrucción de n+una dirección	N+one address instruction	Formato de instrucción que incluye n+1 direcciones, siendo, por lo general, la dirección n+1 la que indica la instrucción que ha de ejecutarse a continuación.
Instrucción de transferencia	Transfer instruction	Instrucción que copia datos de una parte de memoria a otra. Instrucción de bifurcación que sirve para transferir el control de una parte de un programa a otro.
Instrucción en lenguaje fuente, Sentencia	Statement	También se emplea en cualquier expresión que se pueda introducir en un compilador, incluyendo las sentencias de comentarios y las pseudo instrucciones que controlan el funcionamiento de la compilación. Una vez compiladas, las sentencias generalmente dan por resultado varias instrucciones en código de máquina.
Instrucción no operativa	Waste instruction	Instrucción que se incorpora a un programa para asegurar la secuencia de otras instrucciones de dicho programa, preparando cambios futuros en el mismo o cumplimentando ciertas condiciones necesarias en éste (completar un bloque de instrucciones, esperar determinado tiempo, por ejemplo), pero sin especificar operación alguna.
Instrucción nula	Null instruction	Sinónimo de no operación.
Instrucción sin dirección	No-address instruction	Instrucción que no necesita especificar una dirección en la memoria.
Integrador	Integrator	Circuito o dispositivo cuya función de salida es proporcional a la integral de la función de entrada respecto a una variable determinada.
Interacción	Interaction	En sistemas que trabajan en tiempo compartido, unidad de tipo básico que se emplea para medir la actividad del sistema.
Interacción	Iteration	Repetición de un grupo de instrucciones.
Interactivo, Conversacional	Conversational	Sistema en el que el computador puede reaccionar después de cada sentencia del usuario o de entrada.

Término en español	Término en inglés	Definición
Intercambio de señales de control	Handshaking	Técnica de sincronización de comunicación entre dos equipos (generalmente módems o terminales).
Intercepta-ción premeditada	Willful intercept	Acción por la que se interceptan mensajes destinados a estaciones terminales que están experimentando dificultades o anomalías de funcionamiento en el equipo o en la línea.
Interpola-ción	Interleaving	Asignación de emplazamientos sucesivos de memoria a diferentes módulos físicos de memoria.
Intérprete	Interpreter	Programa de traducción empleado para ejecutar sentencias en lenguajes de alto nivel una a la vez.
Interrupción	Interrupt	Señal de atención enviada por un dispositivo de E/S o circuito a una ***UPC*** para romper el funcionamiento normal de la rutina que se está ejecutando. Señal que origina la pausa.
Intervalo, Separación	Gap	Intervalo de espacio o tiempo que separa palabras, registros o archivos en una cinta magnética.
Jerarquiza-ción, Anida-miento	Nesting	Término relativo a una rutina o subrutina que contiene una estructura similar a sí misma.
Jerarquizar	Rank	Acción de disponer series según su importancia.
Juego de caracteres del lenguaje	Language character set	Conjunto de caracteres definidos para utilizar en un lenguaje de programación.
Juego de instrucciones	Instruction set	Repertorio o colección de instrucciones que constituyen el lenguaje de programación de una computadora o sistema de programación.
K	K	Símbolo equivalente a 10^3.
KIPS	KIPS	Mil instrucciones por segundo.
Lápiz fotosensible	Pen Light	Dispositivo de entrada para una pantalla de video que registra la emisión de luz en el punto de contacto con la pantalla. La relación de temporización con el comienzo del barrido determina la posición aproximada en la pantalla.
Lazo o bucle de corriente	Current loop	Medio de entrelazar vías de datos en presencia o ausencia de corriente en un cable bifilar.
Lector de cinta*	Tape reader	Dispositivo electromecánico cuya finalidad es detectar las perforaciones representativas de la información codificada existente en una cinta de papel. Dispositivo capacitado para detectar la información registrada en una cinta magnética.

Término en español	Término en inglés	Definición
Lector óptico de códigos de barras	Optical-bar code reader	Dispositivo que lee por medios ópticos la información codificada mediante barras de distintos anchos impresas o pegadas en documentos u otros artículos.
Lectura de salida	Read-out	Lectura realizada por la memoria interna y transferida a un sistema de almacenamiento de tipo externo.
Lectura dispersa	Scatter-read	Proceso consistente en distribuir en diversas áreas de la memoria, los datos procedentes de un solo registro de entrada o bloque.
Leer	Read	Acción de obtener datos almacenados en memoria.
Leer, Captar, Detectar	Sense	Acción de leer las perforaciones existentes sobre una tarjeta perforada. Acción de captar un impulso eléctrico que circula por un circuito. Detectar una condición que reúna unas características determinadas.
Lenguaje	Language	Conjunto de reglas y símbolos estructurados de forma que la combinación de éstos ofrece un significado específico de comunicación.
Lenguaje algorítmico, ALGOL	Algorithmic Language	Lenguaje simbólico de carácter científico.
Lenguaje científico	Scientific language	Lenguaje proyectado para escribir programas científicos o matemáticos.
Lenguaje común para actividades comerciales	COBOL	Lenguaje de alto nivel simbólico de macroinstrucciones, destinadas al tratamiento de problemas esencialmente comerciales.
Lenguaje de alto nivel	High-level language	Lenguaje de programación orientado a la resolución de problemas "fácil" de usar por el usuario final.
Lenguaje de máquina	Machine language	Instrucciones escritas en código de máquina que una computadora puede obedecer directamente sin necesidad de traducción.
Lenguaje de programación algorítmico	ALGOL	Lenguaje de alto nivel que dispone de una estructura conocida como «exenta de contexto».
Lenguaje de programación APL	APL	Lenguaje de programación de alto nivel inventado por Iverson que se emplea para programación interactiva algorítmica.
Lenguaje fuente	Source language	Lenguaje sobre el cual opera un programa traductor.
Lenguaje natural	Natural language	Lenguaje cuyas reglas reflejan y describen el uso común, más que el uso prescrito.
Lenguaje objeto	Object language	Lenguaje o juego de instrucciones codificadas al cual se traduce un lenguaje fuente, valiéndose de un compilador.

Término en español	Término en inglés	Definición
Lenguaje objeto, Lenguaje máquina	Object language	Lenguaje comprensible por la máquina, el cual resulta generalmente de la traducción de un lenguaje simbólico. Está formado por una serie secuencial de instrucciones en lenguaje máquina.
Lenguaje orientado a problemas	Problem-oriented language	Lenguaje especialmente diseñado para su aplicación a un tipo específico de problemas y cuya utilidad en cualquier problema de otras características es prácticamente nula.
Lenguaje PL/I	PL/I	Lenguaje de programación simbólico que combina las características típicas de los lenguajes administrativos con las de los lenguajes científicos.
Lenguaje resultante, Lenguaje objeto	Target language	Lenguaje al cual se traduce una sentencia o programa.
Limitar	Slice	Acción de eliminar las partes de una forma de onda que quedan fuera de unos límites de amplitud dados en el mismo lado del eje cero.
Línea de retardo	Delay line	Circuito en cuyo diseño específico intervienen unos componentes que hacen posible la introducción de un retardo en la transmisión de la señal.
Línea de retardo	Line delay	Dispositivo concebido para retardar la transmisión de una información determinada.
Líneas por minuto	LPM (Lines per minute)	Unidad que determina la velocidad de impresoras.
Listado	Listing	Documento impreso, generalmente sobre papel continuo, de la información obtenida por la computadora.
Llamada	Call	Instrucción utilizada para transferir la secuencia de ejecución del programa a una subrutina o subprograma.
Llave, Clave, Tecla	Key	Conjunto de caracteres de un registro o de un conjunto de registros que se utilizan para identificar a éstos. Cualquier tecla, conmutador, etc., que puede cerrar o abrir un circuito o generar un movimiento mecánico.
Localización de errores, Investigación de averías	Trouble-shooting	Investigación o búsqueda de los errores de un programa o de la causa de algún fallo de máquina.
Lógica combinacional	Combinational logic	Circuito que realiza funciones de lógica de Boole sin contar con memoria.
Lógica de inyección, Inyección	Integrated-injection logic (I^2L)	Tecnología bipolar para fabricación de circuitos con integración a gran escala (LSI). Se caracteriza fundamentalmente por su bajo consumo.

Término en español	Término en inglés	Definición
integrada, Lógica I2L		
Longitud de cadena	String length	Número de bits o caracteres que integran una cadena.
Longitud de palabra	Word length	Número de bits o caracteres que integran una palabra de máquina.
Lote	Batch	Colección de documentos, registros o transacciones agrupadas con la finalidad de procesarlos como una sola unidad. Durante este proceso, no es posible realizar comunicaciones interactivas entre el programa y el usuario. En algunos sistemas operativos se permiten también instrucciones de administración de recursos en lotes.
Lote, Archivo de tarjetas*	Pack	Colección de tarjetas perforadas relativas a un mismo asunto y que generalmente se ordenan siguiendo un criterio específico. Empaquetar, condensar. Acción de situar más de una unidad de información en una sola unidad de almacenamiento en orden a un ahorro de espacio de memoria.
Macro instrucción	Macro instruction	Instrucción individual que se escribe como parte de un lenguaje fuente y que cuando se compila en un programa en código de máquina, genera varias instrucciones en código de máquina.
Manipulación por desplazamiento de frecuencia	Frequency-shift keying (FSK)	Modulación de una frecuencia de una señal portadora que se lleva a cabo modulando una señal que varía entre un número fijo de valores discretos.
Mantenimiento previsto o programado	Scheduled maintenance	Mantenimiento, generalmente preventivo, que con carácter más o menos periódico se efectúa, según un programa previamente establecido.
Manufactura auxiliada por computadora	CAM	Programa de aplicación que ayuda en la manufactura.
Máquina	Machine	Dispositivo o equipo capaz de realizar por si solo alguna función.
Marca	Mark	Carácter empleado para identificar el final de un conjunto de datos.
Marca de dirección	Address mark	Código especial de 8 bits empleado en el comienzo de una zona específica de la pista de un disco.
Marca de palabra	Word mark	Símbolo utilizado para indicar el principio o fin de una palabra de máquina.
Mega	Mega	Prefijo que indica 10^6.
Memoria	Memory	Dispositivo en el que se pueden introducir datos, donde se mantienen y donde pueden retirarse posteriormente. En sentido genérico, dícese de cualquier dispositivo capaz de almacenar datos.

Término en español	Término en inglés	Definición
Memoria alterable	Alterable memory	Sistema de almacenamiento en el que puede escribirse.
Memoria anexa	Bump	Memoria que no puede ser direccionada directamente por el programador.
Memoria de acceso aleatorio, Memoria RAM	Random-access memory (RAM)	Almacenamiento concebido para proporcionar un tiempo de acceso constante para cualquier posición direccionada, cualquiera que sea la posición previamente direccionada.
Memoria de almacenamiento descendente	Push down store	Memoria que constituye, en cuanto a los componentes de la máquina, la aplicación o representación práctica de una lista de desplazamiento descendente. Contrasta con el término "Push up store".
Memoria de burbujas*	Bubble memory	Dispositivo de almacenamiento que utiliza dominios magnéticos de tipo microscópico en un substrato de granate de aluminio.
Memoria de contenido direccionable	CAM	Memoria asociativa direccionada por el contenido en lugar de por la posición.
Memoria de dos núcleos por bit*	Two-core-per-bit store	Concepto que describe una memoria en la que cada dígito binario se representa por dos núcleos magnéticos.
Memoria de mercurio*	Mercury memory	Dispositivo de almacenamiento en el que la información se retiene haciendo recircular señales en una línea de retardo de mercurio.
Memoria de N núcleos por bit*	N-core per-bit store	Memoria en la que cada unidad elemental de información magnética está integrada por "n" núcleos magnéticos.
Memoria de sólo lectura (ROM) de control	CROM	Memoria ***ROM*** usada para almacenar programas de control de dispositivos.
Memoria de sólo lectura, Memoria pasiva, Memoria inalterable, Memoria ROM	Read-only storage (***ROM***)	Dispositivo capaz de retener datos, los cuales no se pueden alterar por instrucciones de programa.
Memoria de acceso cero	Zero access storage	Memoria cuyo tiempo de latencia o de espera es muy breve. Este término, utilizado frecuentemente en otro tiempo, está perdiendo aceptación en la actualidad, ya que implica un concepto falso o erróneo.
Memoria de trabajo	Scratch pad memory	Área de la memoria reservada para los trabajos intermedios.
Memoria dinámica	Dynamic memory	Memoria ***RAM*** que utiliza circuitos dinámicos; cada bit se almacena en forma de

Término en español	Término en inglés	Definición
		carga en un transistor que se debe renovar constantemente.
Memoria estática	Static memory	Memoria ***MOS*** que utiliza un multivibrador biestable como elemento de almacenamiento. No necesita regrabado y no requiere reloj; además tampoco pierde su contenido siempre y cuando no se corte la alimentación.
Memoria intermedia	Buffer	Memoria, sección de memoria o registro que recoge la información procedente de una memoria auxiliar, o más generalmente, de un dispositivo de E/S, reteniéndola para transferirla oportunamente a la memoria interna o principal de la computadora. En un sentido amplio, dícese de cualquier memoria destinada a retener información durante un tiempo determinado, hasta el momento de la transferencia adecuada.
Memoria intermedia de textos	Text buffer	Memoria provisional que contiene única y exclusivamente el texto de los mensajes.
Memoria matricial	Matrix store	Memoria cuyos elementos están dispuestos en forma tal que el acceso a cualquier posición exige el empleo de dos o más coordenadas.
Memoria no volátil, Memoria estable	Non-volatile memory	Medio de almacenamiento cuyo contenido no se pierde cuando se corta la corriente y se encuentra disponible cuando retorna la alimentación.
Memoria pasiva (ROM) programable	***EPROM***	Típicamente se refiere a las memorias programables de sólo lectura (***PROM***) que pueden borrarse por exposición a los rayos ultravioleta.
Memoria principal	Main memory	Memoria interna de una computadora; es decir, almacenamiento de acceso inmediato, a diferencia de cualquier almacenamiento auxiliar que pueda formar parte del sistema de la computadora.
Memoria volátil	Volatile storage	Sistema de memoria en que los datos almacenados se pierden cuando se desconecta la corriente que alimenta al sistema.
Micro	Micro	Prefijo que indica 10^{-6}. Abreviación de microcomputadora.
Microinstrucción	Microinstruction	Instrucción que forma parte de una sección de microcodificación. Uno de los pasos constitutivos de una macroinstrucción.
Microprocesador expandible	Bit-slice microprocessor	Componente que efectúa una separación de n bits en una ***UPC*** tradicional en la que normalmente n = 4. Un microprocesador expandible contiene todos los elementos de la ***UPC***, incluyendo los

Término en español	Término en inglés	Definición
		multiplexadores, desplazadores, ALU, registros y acumuladores.
Mili	Milli	Prefijo que indica 10^{-3}.
Modulador/demodulador, Modem	Modem	Dispositivo que permite efectuar la transmisión de datos a grandes distancias montando la información en otra señal conocida como portadora.
Monitor	Monitor	Cualquier dispositivo que examina el estado de un sistema para indicar cualquier desviación que se produzca con respecto a las condiciones de funcionamiento. Pantalla de video.
MOS complementario	CMOS	Tecnología que se caracteriza por poseer muy bajo consumo. Esta tecnología requiere un canal tipo P y un transistor de canal tipo N, tiene la velocidad e integración intermedias entre NMOS y PMOS. Dispone de unas características ideales en cuanto a inmunidad al ruido.
Muestreo	Sampling	Proceso consistente en registrar el valor de una variable a determinados intervalos.
Muestreo	Strobe	Señal de selección que se activa cuando los datos en una barra (bus) son correctos.
Muestreo y retención	Sample and hold	Función realizada por un circuito analógico que captura y retiene una señal para luego ser convertida por un convertidor analógico-digital.
Multiplexado por división de tiempo	Time-division multiplexing	Sistema por el que un canal se pone a disposición de un número determinado de dispositivos terminales. Cada uno de estos dispositivos ocupa el canal, para la transmisión de datos, durante cortos períodos y a intervalos regulares.
Multiplexor	Multiplexor	Dispositivo de control de comunicaciones que permite enviar/recibir por una línea física varias señales a la vez.
Multiprocesador	Multiprocessor	Procesador central que contiene dos o más unidades aritméticas independientes además de su correspondiente lógica de control.
Multiproceso	Multiprocessing	Proceso mediante el cual se emplean varios procesadores para el tratamiento de una sola transacción.
Multiprogramación	Multiprogramming	Técnica que permite tratar simultáneamente varios programas diferentes, o fragmentos de programas, ejecutando la intercalación de los mismos bajo el control de un programa supervisor.
Nanosegundo	Nanosecond	Prefijo que denota la mil millonésima parte de un segundo (10^{-9} segundos).

Término en español	Término en inglés	Definición
Neutralización, Inhabilitación	Disable	Supresión o anulación de un dispositivo.
NO-Y	NAND	Puerta NO-Y. (Véase NOT)
NO-Y	NOT-AND	Operación lógica que cumple con la regla de que la salida C es verdadera si cualquiera de las entradas es falsa.
Núcleo de bobinado magnético*	Tape wound core	Núcleo magnético constituido por un bobinado de cinta ferromagnética.
Núcleo, Ferrita*	Core	Pequeños toroides magnéticos de ferrita empleados para almacenar información de un bit. Cilindro en cuyos laterales se incluyen unos rebordes con la finalidad de permitir que se enrolle una cinta.
Número binario	Binary number	Cantidad representada en un código binario; es decir, mediante un conjunto de unos y ceros.
Número de serie	Serial number	Número que se adjunta a una información para reconocer la posición que dicha información ocupa en una serie ordenada de ellas. Número de serie de un programa.
Octal	Octal	Término relativo al sistema de representación numérica con base ocho (en desuso pues se prefiere la base 16).
Octeto	Octet	Carácter o posición compuesta de ocho bits.
Octeto, byte	Byte	Término que representa una porción medible de dígitos binarios consecutivos.
Operación	Operation	Acción determinada por medio de la cual se obtiene un resultado de un operando. Acción definida por una sola instrucción de la computadora. Acción definida por un solo elemento lógico.
Operación unitaria	Unary operation	Operación que se ejecuta sobre un solo operando.
Operando	Operand	Elemento o dato de una operación del cual se obtiene el resultado por medio de acciones definidas.
Orden inferior	Low order	Relativo a la importancia o significado atribuido al dígito situado en el extremo derecha de un número expresado en notación posicional.
Ordenación	Ordering	Clasificación o disposición en secuencia o en serie.
Palabra	Word	Unidad lógica de información que puede tener cualquier número de bits, pero normalmente es de 4, 8 o 16.
Palabra clave	Keyword	Palabra característica de un archivo, que se utiliza para encontrar su contenido mediante un significado.

Término en español	Término en inglés	Definición
Palabra de control	CW	Palabra almacenada en memoria que se utiliza para controlar las prioridades que se han de observar en la selección de las operaciones a desarrollar cuando se emplean las técnicas de tiempo compartido.
Palabra numérica	Numeric word	Palabra formada por dígitos y, posiblemente, por caracteres de espaciado y especiales.
Palanca de mando	Joystick	Palanca normalmente de tipo vertical que puede inclinarse en cualquier sentido para indicar dirección o movimiento. Se utiliza para desplazar un punto en pantalla. Muy utilizada en juegos.
Parada	Halt	Estado en el que una computadora deja de funcionar o se detiene.
Paralelo, en paralelo	Parallel	Método de tratamiento simultáneo de todos los elementos de un artículo o unidad de información.
Parámetro	Parameter	Valor fijo que se concede a una variable, durante el proceso de un problema específico por medios automáticos.
Parche	Patch	Grupo de instrucciones que se agregan a una rutina para corregir una equivocación.
Pasada	Pass	Paso de una cinta magnética por las cabezas de lectura.
Paso entre filas, Paso*	Row pitch	Distancia entre orificios perforados en sentido longitudinal a lo largo de una cinta de papel perforada; medida entre los centros de dos posiciones consecutivas.
Paso entre pistas, Separación entre pistas*	Track pitch	Distancia que separa dos pistas contiguas.
Pérdida de alimentación*	Misfeed	Funcionamiento defectuoso de un equipo cuyo sistema de alimentación de información ha de efectuarse mediante tarjetas perforadas, por mala perforación, mala lectura o deterioro de la tarjeta.
Pérdida de transmisión	Transmission loss	Sinónimo de atenuación.
Pérdidas acumuladas	Walk down	En una memoria que funciona incorrectamente, los impulsos excitadores parciales o los impulsos de dígito sucesivos provocan un proceso magnético irreversible en una celda magnética. Este proceso se conoce como pérdidas acumuladas o pérdidas de información.
Perforación de zona*	Zone punch	Perforación realizada en las filas 12, 11 y 0 de una ficha perforada. Perforación distinta a la numérica.

Término en español	Término en inglés	Definición
Perforación en cadena*	Lace	Operación consistente en perforar todas las posiciones de una columna en una ficha.
Perforación en serie*	Gang punch	Acción de perforar información en una tarjeta de datos.
Perforación X*	X punch	Orificio que se perfora en la posición X de una ficha perforada (por lo general, la segunda fila empezando por la parte superior).
Perforación Y*	Y-punch	Orificio que se perfora en la posición Y de una ficha (por lo general, en la fila superior). Posición Y.
Perforar*	Punch	Acción consistente en efectuar un orificio en una tarjeta perforada o en cinta de papel para su uso.
Periférico	Peripheral unit	Máquina que puede funcionar bajo el control de la computadora. El equipo periférico consta de dispositivos de E/S y de almacenamiento.
Petición automática de repetición	ARQ	Sistema que emplea un código para detección de errores, dispuesto de forma que una señal detectada como errónea, origina automáticamente una petición de retransmisión.
Picosegundo	Picosecond	Milésima parte de un nanosegundo. 10^{-12} segundos.
Pila	Stack	Estructura LIFO que guarda la información en orden cronológico.
Pista*	Track	En un dispositivo de memoria magnética, canal que sirve para registrar datos. Uno de los canales para registrar datos en forma de perforación en una cinta de papel.
Poner a ceros	Zeroize	Restaurar un registro mecánico o electrónico restituyéndolo a su posición o estado cero.
Portadora, Soporte de datos	Data carrier	Medio físico sobre el que se registran los datos de forma que, además de permitir transporte, ha de permitir la interpretación de los mismos por los equipos correspondientes.
Preeditar	Pre-edit	Técnica consistente en realizar una pasada preliminar de edición de los datos de entrada, antes de emplear los datos para un proceso posterior.
Preparación previa	Housekeeping	Conjunto de operaciones realizadas para el funcionamiento correcto del proceso o programa.
Preparación, Puesta a punto	Setup	Serie de operaciones con las que se prepara a las unidades que componen un equipo para su funcionamiento.
Primero en entrar-	FIFO	Estructura en la cual los datos se depositan en un extremo y se extraen del otro.

Término en español	Término en inglés	Definición
primero en salir, Cola de espera, FIFO		Las FIFO se utilizan como uniones para conectar dos dispositivos que trabajan en forma asíncrona y a distintas velocidades.
Probar, Examinar	Test	Acción de examinar un elemento de datos o un indicador para determinar si satisface alguna condición predeterminada.
Procedimiento incorporado	In-stream procedure	Conjunto de instrucciones de control de trabajos que se colocan en la corriente de entrada y que pueden utilizarse un número indefinido de veces durante un trabajo.
Procesador periférico	Peripheral processor	En un sistema de proceso de datos en el que se emplea más de una unidad de proceso, se denomina procesador periférico a aquél que opera bajo el control de otro.
Procesador, Unidad de proceso	Processor	Término general que designa cualquier dispositivo capaz de desarrollar operaciones con datos.
Proceso	Process	Término que indica la ejecución de todas las operaciones de un tratamiento: lectura, compilación, cálculo, etc.
Proceso de datos integrados, IDP	Integrated data processing	Tratamiento de una información conseguido a través de una coordinación general que mejora la eficiencia del trabajo.
Proceso secuencial	Sequential processing	Proceso de los registros de un archivo de datos que se efectúa de acuerdo con una secuencia predeterminada de claves.
Programa	Program/programme	Grupo de instrucciones compuestas con la finalidad de resolver un problema específico mediante una computadora.
Programa almacenado	Stored program	Programa totalmente contenido en memoria y que se puede alterar dentro de ella.
Programa auto inicializado	Self-triggering program	Programa diseñado de forma que su operación comienza automáticamente cuando se carga en la computadora.
Programa cruzado	Cross program	Programa para una computadora A que reside (y se ejecuta) en una computadora B.
Programa de utilidad	Utility program	Programa cuya estructura está dirigida a prestar un servicio en la ejecución de otro programa, generalmente de aplicación. Por ejemplo, un programa de entrada.
Programa de verificación	Test program	Programa especialmente diseñado para demostrar que un equipo está en condiciones de ejecutar un proceso específico.
Programa inicializador	Bootstrap	Programa utilizado para poner en funcionamiento una computadora. Usualmente pone a cero la memoria, pone a punto los dispositivos de E/S y carga el sistema operativo procedente de la memoria ***ROM***, disco (flexible o duro) o casete (en desuso).

Término en español	Término en inglés	Definición
Programa objeto	Object program	Programa obtenido en lenguaje de máquina después de la compilación o ensamblaje de otro simbólico. Es, pues, un programa simbólico traducido a otro de máquina.
Programa patrón	Benchmark program	Programa específico para medir la velocidad de una computadora en una operación bien definida, tal como una transferencia en serie, de multiplicar 8 por 8 bits, etc.
Programa principal	Main program	Parte central de un programa que generalmente transfiere el control a otras subrutinas según la naturaleza de los datos que sé están procesando. Estructura central en la que se ensamblan las diversas secciones o subrutinas.
Programa resultante	Target program	Sinónimo de programa objeto.
Programa subordinado	Background program	En multiprogramación, dícese del programa de baja prioridad que opera cuando el procesador no trabaja con programas preferentes.
Programa, Soporte lógico, Conjunto de programas, software	Software	Conjunto de programas y procedimientos que se incluyen en un equipo de tratamiento de datos y que hace posible la utilización eficaz del mismo.
Programación	Programming	Proceso que comprende el diseño, escritura y prueba de un programa. Arte de reducir a instrucciones detectables por la máquina el plan para la resolución de un problema.
Programas de apoyo	Support programs	Conjunto de programas destinados a apoyar los básicos de un sistema operativo.
Protección de memoria	Memory guard	Dispositivo que forma parte de los componentes físicos de la computadora o de su soporte lógico, que tiene por función impedir que un programa direccione posiciones específicas de la memoria interna.
Pseudo instrucción, Instrucción falsa	Pseudo-instruction	Grupo de caracteres que tienen el mismo formato general que una instrucción de ***UPC***, pero que éste nunca ejecuta como una instrucción real.
Pulso de inicialización	Reset pulse	Impulso de puesta en estado inicial. Impulso cuya finalidad es hacer que un elemento de almacenamiento binario cambie el estado existente en un momento determinado por otro estado fijo.
Punto	Point	Símbolo que separa la parte entera de la fraccionaria de un número. En Europa y otros países se usa la coma.

Término en español	Término en inglés	Definición
Punto de interrupción, Punto de ruptura	Breakpoint	Dispositivo de hardware o software que detiene un programa y permite analizar el estado hasta el punto de la interrupción.
Punto de recuperación	Rerun point	Lugar dentro de una secuencia de instrucciones de un programa que contiene toda la información que se disponga pertinente a la recuperación del mismo. Con ello es posible reconstruir un proceso, si ocurre un error o si es necesario interrumpirlo antes de que éste termine.
Punto flotante	Floating point	Punto que separa la parte entera de la decimal y cuya posición depende del resultado de cada operación.
Raíz, Base	Radix	Número entero adoptado para definir un sistema de numeración.
Rango, Escala	Range	Conjunto de valores que puede tomar una cantidad.
Rastreo, Diagnóstico, Trazado	Trace	Análisis de los resultados obtenidos tras la ejecución de cada instrucción, con el fin de obtener un diagnóstico de funcionamiento directo.
Reanudar	Restart	Acción de retornar a un punto previo de un programa para comenzar de nuevo a continuación de un error o de un fallo de máquina.
Rebasamiento	Overflow	Efecto producido cuando el resultado de una operación aritmética excede la capacidad de valores admitidos para la representación de un número o de un registro.
Rebobinar	Rewind	Acción de colocar de nuevo una cinta magnética en el punto de carga.
Recolección de datos	Data collection	Forma de capturar datos en la que la información se concentra en un punto central antes de procesarse.
Recolector de Basura	Garbage collector	Acción de compactar los espacios libres que se generan en la memoria al ejecutar un programa y desechar partes del código o de las variables.
Reconocimiento de caracteres ópticos	OCR (optical character recognition)	Reconocimiento por medios automáticos de caracteres impresos por procedimientos fotoeléctricos.
Recubrimiento, Traslape	Overlay	Técnica utilizada cuando las necesidades de memoria son superiores a las disponibilidades. Consiste en realizar la carga de rutinas en memorias de alta velocidad procedentes de otras, de forma que las mismas posiciones de memoria se ocupen por diferentes rutinas en momentos distintos.
Recuperación	Retrieval	Localización y aislamiento del material almacenado o Información específica.

Término en español	Término en inglés	Definición
Red	Network	Relativo a cualquier sistema, serie de puntos y sus interconexiones.
Redondear	Round	Alterar el valor de los dígitos en el extremo menos significativo de un número a fin de permitir la eliminación de los dígitos al truncar el número.
Redondeo	Rounding-off	Técnica utilizada para representar números y reducir posibles desviaciones a base de incrementar el tamaño de la cantidad.
Redundancia	Redundancy	Empleo de caracteres o bits adicionales que se agregan a un grupo de datos a fin de proporcionar un medio de comprobación de la exactitud de los datos.
Registrador	Logger	Dispositivo que registra automáticamente las condiciones físicas respecto al tiempo.
Registro	Record	Grupo de elementos de datos relacionados entre sí que son tratados como una sola unidad.
Registro	Register	Números de posiciones de memoria que contienen una información referida a un aspecto específico. Dispositivo hardware, cuya función consiste en retener una información que se ha de tratar a continuación.
Registro de base	Base register	Registro que contiene direcciones de base para referencias de tipo indexado. La dirección final se obtiene por suma de un desplazamiento.
Registro de desplazamiento	Shift register	Registro en que los datos almacenados se pueden someter a un desplazamiento hacia la derecha o hacia la izquierda.
Registro de estado	Status register	Registro utilizado para mantener la información de estado dentro de una unidad funcional, como una ***UPC***, una PIC, etc. Un registro de estado de ***UPC*** típico proporciona indicación de acarreo, rebasamiento, signo, cero e interrupción. Puede también incluir paridad, desinhibición o máscara.
Registro de modificación	Change record	Registro cuya misión consiste en alterar o cambiar la información del registro principal correspondiente.
Registro de operaciones	Operation register	Registro en el que se almacena el código de operación durante el ciclo de operación.
Registro dedicado	Dedicated register	Registro empleado exclusivamente para contener datos específicos.
Regrabación, Reescritura	Rewrite	Retención de datos en una zona de almacenamiento. Registrándolos de nuevo en la posición de que se trate después de leerlos en esa posición.

Término en español	Término en inglés	Definición
Reincorporar a memoria	Rolling	Acción de restaurar en memoria principal programas y datos previamente transferidos desde esta memoria a un almacenamiento auxiliar.
Relación de caracteres erróneos	Character error rate	Relación entre el número de caracteres recibidos correctamente y el número total de los transmitidos.
Relación del rechazo en modo común	CMRR	Ganancia de modo común en amplificadores operacionales.
Rellenar con ceros	Zero fill	Rellenar con caracteres utilizando la representación de cero.
Relleno	Padding	Acción de agregar blancos o caracteres no significativos al final de un registro o bloque, con objeto de darle un tamaño determinado.
Reloj, Generador de impulsos	Clock	Dispositivo que genera una base de tiempos utilizada para proporcionar los impulsos secuenciales básicos para las operaciones de una computadora, cuya forma de trabajo es secuencial.
Rendimiento específico, Productividad	Throughput	Productividad de una máquina, sistema o procedimiento, medidos según un factor de comparación que tenga significado para el proceso de que se trate.
Repertorio, Juego	Repertoire/repertory	Gama de caracteres o de códigos individuales de que se dispone en un sistema de codificación determinado.
Repetidor terminal	Terminal repeater	Repetidor que se utiliza en el extremo de una línea de enlace.
Repetir	Rollback	Ejecutar de nuevo un programa (o parte) en una computadora.
Respuesta afirmativa	Acknowledge	Señal de control utilizada para completar una secuencia de confirmación de una secuencia de intercambio de indicativos y señales de control.
Respuesta automática	Auto-answer	Dispositivo que confiere a las unidades de transmisión de datos, la facultad de responder de forma automática a una llamada recibida.
Restador con dos entradas	Two-input subtractor	Unidad de un componente aritmético con posibilidades de recibir dos señales de entrada que representan un dígito de un número y otro que corresponde al sustraendo (o al dígito de arrastre) y dos señales de salida que representan al dígito diferencia correspondiente y al de arrastre que debe operar con la siguiente posición de número.
Restador con tres entradas	Three-input sutractor	Unidad de resta capaz de recibir tres señales de entrada (una que representa el minuendo, otra el sustraendo y la tercera el dígito de préstamo) y de emitir dos

Término en español	Término en inglés	Definición
		señales de salida (una que representa al dígito de diferencia y otra el préstamo, con el que ha de operarse en la siguiente posición de dígito).
Restaurar	Restore	Colocar o fijar en un contador, un registro, un conmutador o un indicador a un valor o condición previos. Devolver una dirección variable u otra palabra de la computadora a su valor inicial o seleccionado.
Restaurar poner a cero	Reset	En programación, acción de poner a cero un contador o devolver un indicador a alguna condición estable.
Residuo	Remainder	Diferencia entre el dividendo de una división y el producto del divisor por el cociente.
Retardo	Lag	Intervalo existente entre dos estados.
Retardo de propagación	Propagation delay	Tiempo total invertido por una señal en su desplazamiento de un punto a otro de un circuito.
Retorno	Fly back	Período en el cual el punto presentado en una pantalla regresa al principio de una línea.
Reubicar	Relocate	Acto de modificar automáticamente las instrucciones de un programa, que se lleva a cabo para permitir la carga y ejecución del programa en cualquier área de memoria.
Robo de ciclos	Cycle stealing	Método por el cual otra ***UPC***, dispositivo de acceso a memoria u cualquiera otro gana acceso al bus del microprocesador. En lugar de permitir que la ***UPC*** use el bus durante un ciclo para la búsqueda y decodificación de datos, se cede al otro dispositivo que lo solicitó.
Rollo, Bobina	Reel	Bobinado de cinta de papel, montado en un soporte de cartón o plástico.
Rótulo, Distintivo	Tag	Símbolo anexo a un elemento, utilizado para su identificación.
Rótulo, Etiqueta	Label	Conjunto de caracteres de un registro colocados en el mismo para su identificación. Identificación visible que explica el contenido de un almacenamiento externo.
Rueda de tipos	Type wheel	Componente de una impresora de líneas donde los caracteres están contenidos en una rueda que se posiciona por giros en sentido vertical.
Ruido	Noise	Señales extrañas, generalmente no deseadas, que surgen circunstancialmente en cualquier parte de un sistema de transmisión de datos.
Ruido delta	Delta noise	Uno de los ruidos que se producen en el proceso de lectura de una celda magnética y cuya participación se hace patente sobre

Término en español	Término en inglés	Definición
		la tensión inducida del detector de conducción.
Rutina	Routine	Parte de un programa, generalmente diseñado para una aplicación específica, cuyo empleo suele ser muy frecuente.
Rutina de utilidad	Utility routine	Tipo de rutina utilizada para colaborar en el proceso de una computadora y para tratar las operaciones de máquina necesarias para el proceso de datos, pero que no contribuye directamente a la producción de resultados.
Rutina de verificación	Test routine	Rutina especialmente concebida para demostrar que un equipo está en condiciones de efectuar un determinado proceso.
Rutina objeto	Object routine	Rutina en lenguaje de máquina que constituye la salida de la traducción del lenguaje fuente. Rutina de ejecución.
Rutina residente	Resident routine	Rutina que se encuentra registrada permanente en memoria.
Salida	Output	Resultados producidos por una computadora. Transferencia de información desde una ***UPC*** a un dispositivo de salida.
Salida impresa	Hard-copy	Registro impreso en papel que se obtiene en la impresora.
Saltar	Skip	Omitir, saltar o eludir una o más instrucciones en una secuencia de instrucciones.
Salto	Jump	Desviación con respecto a la secuencia normal de ejecución de las instrucciones en una computadora.
Salto de papel	Paper throw	Movimiento del papel por la impresora que difiere del avance normal del papel para su impresión y se efectúa a una velocidad mayor que la del espaciado de una sola línea. Equivale a una alimentación de hoja o de línea (FF o LF)
Salto incondicional	Unconditional jump	Salto que se produce en la ejecución de la secuencia normal de unas instrucciones al aparecer una instrucción de transferencia incondicional.
Secuencia	Sequence	Conjunto de términos o instrucciones que se han colocado o dispuesto en un orden definido.
Secuencia de exploración	Scanning rate	Frecuencia de muestreo de una computadora.
Secuencia de números aleatorios	Random-number sequence	Serie de números imprevisibles producidos por cambios, que satisface una o más de las pruebas de aleatoriedad.
Segmentar, Segmento	Segment	Concepto que expresa la idea de dividir un programa en partes más o menos modulares, para almacenarlo en memoria fragmentado. Para ejecutar el programa no es necesario mantenerlo completo durante el tiempo que dure el proceso.

Término en español	Término en inglés	Definición
Seleccionar	Select	Elegir, según un criterio, uno o más procesos de trabajo o unos dispositivos que se adoptan a un método de trabajo.
Selector	Selector	Dispositivo que efectúa una prueba para determinar la presencia de condiciones específicas e inicia las operaciones apropiadas, de acuerdo con el resultado de la prueba.
Semántica	Semantics	Ciencia que estudia el significado asignado a la construcción de un lenguaje.
Semidúplex	Half-duplex	Técnica utilizada en comunicaciones en la que los datos se pueden transmitir en una sola dirección a la vez.
Semisuma	Half-adder	Circuito sumador que no toma en cuenta el acarreo (arrastre) anterior.
Señal de parada	Stop signal	Señal utilizada para poner un receptor en reposo preparándolo para recibir la señal siguiente.
Señal de salida cero	Zero output signal	Salida que proporciona una localidad de memoria en el estado o condición cero cuando se le aplica un pulso de lectura.
Señalizador	Flag	Carácter que señala alguna condición, como puede ser el final de una palabra.
Señalizador de cero	Zero flag	Señalización del estado de la ALU que indica que la operación anterior es cero.
Serie	Serial	Concepto que se aplica a operaciones aritméticas, transferencias, transmisión de datos, etc., y que indica que éstas se ejecutan dígito a dígito o carácter a carácter.
Símbolo	Symbol	Cualquier carácter, conjunto de caracteres o cifra, aceptados convencional o arbitrariamente como representativos de alguna cantidad, instrucción, etc.
Simplex, Unidireccional	Simplex	Canal de comunicaciones que permite la transmisión en un sentido solamente.
Simulador	Simulator	Sistema diseñado para efectuar la simulación de un proceso en tiempo real.
Sincronización, Temporización	Timing	Proceso de cálculo y regularización de unas operaciones con respecto a un ritmo de tiempo.
Sincronizador	Synchronizer	Dispositivo de almacenamiento que actúa como memoria intermedia para contrarrestar los efectos de la transmisión de datos entre dispositivos que operan a velocidades distintas.
Sintaxis	Syntax	Conjunto de reglas que rigen la escritura de las sentencias o expresiones del lenguaje fuente.
Sistema de barras cruzadas	Crossbar system	Sistema de conmutación de líneas (telefónicas) que utiliza conmutadores de barras cruzadas para identificar el número que se marca.

Término en español	Término en inglés	Definición
Sistema de desarrollo	Development system	Sistema microcomputador que dispone de todos los elementos necesarios para el desarrollo de hardware (electrónica) y software (programas) de un microprocesador dado. Como mínimo incluye: ***UPC***, memoria, pantalla (o teletipo), impresora, almacenamiento masivo, programador de ***PROM*** y emuladores.
Sistema multisecuencial	Multisequential system	Sistema que puede intercalar instrucciones pertenecientes a programas o secuencias diferentes, ejecutándolas de una sola vez.
Sistema operativo	Operating system	Conjunto de rutinas encargadas de la supervisión de las operaciones de un sistema de computación (carga, situación en memoria, sucesión de operaciones, ensamblaje, compilación, etc.). El sistema operativo, en cada caso, introduce automáticamente el programa apropiado.
Sistema operativo en disco	DOS	Sistema operativo que integra funciones en archivos de disco, tales como: archivar, asignación de espacio automática, y todo lo relacionado con la administración del sistema.
Situación de emergencia	Fallback	Condición que hace su aparición en el proceso, en la que se hace preciso emplear funciones especiales como sustitutos totales o parciales del sistema que funcionan defectuosamente.
Sobrecarga	Overload	Exceso de mensajes que se producen durante un momento determinado en las líneas de comunicación debidos a la transmisión simultánea de datos a una computadora.
Sobrescribir, Sobregrabar	Overwrite	Técnica consistente en grabar información en una posición de memoria, destruyendo la información que contenía anteriormente.
Solapamiento	Overlapping	Término que designa a toda técnica o dispositivo que permite realizar varias informaciones al mismo tiempo.
Sondeo	Polling	Técnica empleada para controlar las líneas de comunicación. Para ello, el dispositivo de control emite una señal a cada uno de los terminales conectados, estableciendo comunicación cuando el interrogado tiene mensaje para transmitir.
Soporte lógico inalterable	Firmware	Programa depositado en memoria de sólo lectura (***ROM***).
Subconjunto	Subset	Conjunto o grupo de elementos que guardan determinadas relaciones entre sí y

Término en español	Término en inglés	Definición
		forman parte de un conjunto más importante o de jerarquía superior.
Subrutina	Subroutine	Parte de un programa que ejecuta una parte o sección lógica de las funciones generales del programa y que está disponible siempre que se necesite ese juego específico de instrucciones.
Subrutina de dos niveles	Two-level subroutine	Subrutina que contiene otra subrutina dentro de su propia escritura.
Suma	Sum	Resultado obtenido al añadir a un número (addend: primer sumando) otro número (augend: incremento).
Suma de verificación	Checksum	Resultado obtenido de la suma de los miembros que integran una serie de datos y que generalmente se añade al final de la misma, con la finalidad de detectar la presencia o ausencia de errores.
Sumador	Adder	Dispositivo cuya salida representa la suma de las cantidades que constituyen sus entradas.
Sumador con tres entradas	Three-input adder	Unidad sumadora capaz de recibir tres señales de entrada (una que representa el primer sumando, otra el segundo y la tercera el dígito de acarreo) y de emitir dos señales de salida (una representa el dígito suma y otra el dígito que de acarreo para la operación siguiente).
Sumador de dos entradas	Two-input adder	Unidad de un componente aritmético con posibilidad de recibir dos señales de entrada que representan un dígito de un número y otro que corresponde al primer sumando (o al dígito de arrastre) y dos señales de salida que representan el dígito suma correspondiente y el de arrastre que debe operar en la siguiente operación.
Sumador total, totalizador	Full adder	Unidad encargada de formar el número suma de otros dos números tomando en cuenta el acarreo (arrastre) precedente y generando el siguiente si lo hay. La representación de estos números es a base de señales o impulsos eléctricos aplicados en el momento de la entrada de la información.
Sumador, Sumador analógico	Summer	Dispositivo con dos o más entradas analógicas variables y otra de salida, que es la suma de las de entrada.
Sumar y/o restar horizontalmente	Crossfoot	Dícese de la operación consistente en sumar varios campos horizontales de información numérica para verificar o totalizar.
Supresor de impresión,	Stunt box	Parte de un teleimpresor que codifica las señales empleadas para controlar el

Término en español	Término en inglés	Definición
Caja reguladora		funcionamiento de la máquina, en contraste con la información que se ha de imprimir.
Sustracción	Subtraction	Operación aritmética en la que un operando (sustraendo) se resta de otro (minuendo) para formar la diferencia.
Tabla	Table	Conjunto de datos en que cada unidad elemental de los mismos se identifica por una clave: Los datos están dispuestos en forma apropiada para una fácil consulta.
Tabla de consulta	Look-up table	Colección de datos dispuestos en forma adecuada, tendientes a una fácil consulta. Se almacenan en posiciones ordenadas en secuencia que facilitan la búsqueda.
Tabla de decisión lógica, Tabla de verdad	Truth table	Tabla que describe una función lógica mediante el listado de todas las combinaciones lógicas posibles de valores de entrada y la indicación de los valores verdaderos de salida que corresponden a cada combinación.
Tabla de estados, Tabla de transición de estado	State table	Lista de salidas de un circuito lógico basado en las entradas y en previas salidas. Dicho circuito tiene memoria y no puede ser descrito por una simple tabla de verdad.
Tablero de conexiones	Pinboard	Cuadro de control en el que las interconexiones se realizan mediante conectores desprovistos de cable y en forma de clavija.
Tabular	Tabulate	Acción de ordenar datos disponiéndolos en forma de tabla. Acción desarrollada para imprimir totales.
Tambor	Drum	Memoria magnética giratoria que utiliza la superficie de un cilindro (en desuso).
Tarea	Task	Unidad básica de multiprogramación bajo el programa de control. Ejecución de uno o más procedimientos por parte de un solo flujo de control.
Tarjeta	Card	Unidad interna enchufable que contiene los conductores y componentes de los circuitos impresos.
Tarjeta	Board	Placa de baquelita o de fibra de vidrio utilizada para montar los circuitos integrados. Las interconexiones pueden ser enrolladas, soldadas o impresas.
Tarjeta perforada*	Card	Unidad de almacenamiento de información en forma de tarjeta, destinada a ser perforada según un código establecido de orificios. Los orificios se efectuan en posiciones codificadas, previamente definidas.

Término en español	Término en inglés	Definición
Teclado	Keyboard	Grupo de teclas que se utiliza para introducir información en un sistema computacional.
Teleproceso	Telepro-cessing	Término registrado por IBM que se emplea para describir sistemas en los que se conectan localidades distantes a una computadora central por medio de circuitos de transmisión de datos.
Telescritor*	Teletype-writer	Dispositivo terminal telegráfico, parecido a una máquina de escribir, que se emplea para transmitir y recibir mensajes en un sistema de comunicaciones telegráficas.
Teletipo®*	Teletype®	Dispositivo diseñado para mensajes a distancia y recibirlos en forma impresa. La transmisión se efectúa mediante teclado o por medio de cinta de papel perforada.
Télex*	Telex	Servicio automático de intercomunicación que se emplea para la comunicación entre abonados utilizando un equipo telegráfico, tal como los teleimpresores (en desuso reemplazado por el FAX).
Terminación anormal del programa	Abort	Proceso de interrumpir un programa de forma ordenada para devolver el control al operador o al sistema operativo.
Terminal	Terminal	Dícese de cualquier punto en el que se pueden introducir o extraer datos de un sistema de comunicación de datos. También se conoce como terminal de datos.
Texto	Text	Elemento de información de cualquier mensaje, excluyendo aquellos caracteres o bits necesarios para facilitar su transmisión.
Tiempo compartido	Time-share	En términos generales, dícese de la utilización de un dispositivo, equipo u computadora, por dos o más aplicaciones entremezcladas entre sí a las que se les asigna un tiempo finito de utilización en forma cíclica.
Tiempo de acceso	Access time	Tiempo empleado para buscar una palabra en un dispositivo externo (memoria disco flexible, disco duro, etc.).
Tiempo de acceso	Time access	Tiempo invertido en la búsqueda de una información situada en memoria y en su transferencia a la unidad aritmética y lógica (ALU) de la unidad de procesamiento central (***UPC***).
Tiempo de búsqueda	Search time	Tiempo empleado para localizar un registro determinado en un sistema de almacenamiento de datos.
Tiempo de ciclo	Cycle time	Tiempo total empleado por un dispositivo (normalmente memoria) para completar su ciclo y estar disponible de nuevo.

Término en español	Término en inglés	Definición
		Usualmente el tiempo de acceso es menor que el tiempo de ciclo y a veces igual.
Tiempo de deceleración	Deceleration time	Tiempo transcurrido entre la lectura o escritura de un registro sobre una cinta magnética y el momento en que ésta se detiene.
Tiempo de declinación	Decay time	Período de tiempo que tarda una tensión o corriente alterna en disminuir a un 10% de su valor máximo.
Tiempo de ejecución	Execution time	Tiempo requerido para la ejecución de instrucción incluyendo la búsqueda, codificación y ejecución.
Tiempo de ejecución, Tiempo de proceso	Run time	Tiempo durante el cual se procesa un programa. Contrasta con "'tiempo de compilación".
Tiempo de escritura	Write time	Intervalo que transcurre entre el instante en que comienza la transcripción a un dispositivo de almacenamiento y el instante en que se termina.
Tiempo de espera	Waiting time	Tiempo que transcurre entre el momento en que la unidad de control ordena una transferencia de datos hacia o desde la memoria del sistema, y el preciso instante en que ésta comienza.
Tiempo de inactividad	Off-time	Tiempo que transcurre cuando una computadora no está programada para su uso, mantenimiento, reparación, o modificación.
Tiempo de instrucción	Instruction time	Lapso en el que se extrae una instrucción de la memoria principal de una computadora.
Tiempo de lectura	Read time	Tiempo empleado en localizar datos en una sección de memoria y transferirlos a una unidad aritmética.
Tiempo de no utilización	Unused time	Tiempo durante el cual el equipo está desconectado y fuera de servicio.
Tiempo de palabra	Word time	Tiempo necesario para transferir una palabra desde una posición de memoria a otra o desde un dispositivo de almacenamiento a otro.
Tiempo de prueba del programa	Program testing time	Tiempo invertido en la comprobación de un programa de computación.
Tiempo de referencia	Reference time	Tiempo empleado por un impulso para iniciar una acción y que tarda en alcanzar el diez por ciento de su amplitud especificada.
Tiempo de respuesta	Response time	Intervalo comprendido entre el momento en que se somete un trabajo a un sistema de cálculo y el instante en que se obtienen los resultados.

Término en español	Término en inglés	Definición
Tiempo de respuesta	Turna-round time	Tiempo que se necesita para completar una tarea; por ejemplo, para recoger los datos, transcribirlos para su proceso, realizar el cálculo y proporcionar el resultado al usuario.
Tiempo de subida	Rise time	Tiempo necesario para que un impulso eléctrico se eleve desde una a nueve décimas de su valor final.
Tiempo de transferencia	Transfer time	Intervalo comprendido entre el instante en que se inicia la transferencia de los datos a/o desde un dispositivo de almacenamiento y el momento en que se completa dicha transferencia.
Tiempo de *UPC*	CPU time	Período en el que la unidad de procesamiento central se dedica a la ejecución de instrucciones.
Tiempo pasivo	Idle time	Parte del tiempo útil de una computadora que está disponible para su uso, pero no está en funcionamiento activo.
Tiempo productivo, Tiempo activo	Uptime	Período durante el cual un equipo está trabajando o está disponible para efectuar un trabajo.
Tiempo real	Real-time	Término que se aplica al tiempo actual, durante el cual transcurre el proceso. Método de proceso de datos realizado a tal velocidad que virtualmente no transcurre tiempo alguno entre el momento que se formula la consulta y el instante en que se recibe el resultado.
Trabajo	Job	Unidad de código que resuelve un problema, un programa y todas las subrutinas y datos relativos a ella.
Traductor	Translator	Programa que convierte las sentencias escritas en un lenguaje de programación al formato de otro lenguaje de programación. Por ejemplo, de un lenguaje fuente a código máquina.
Traductor de lenguaje	Language translator	Vocablo de tipo genérico con el que se define cualquier ensamblador, compilador u otro tipo de rutina. Admite instrucciones en un lenguaje de programación y produce instrucciones equivalentes en otro, sin cambiar de forma apreciable su significado.
Trampa, Desvío	Trap	Operación de bifurcación que el equipo físico inicia automáticamente al detectar alguna condición anormal durante el proceso de un programa.
Transferencia en paralelo	Parallel transfer	Método de transferencia de datos en el que la transmisión de éstos se realiza de forma simultánea.

Término en español	Término en inglés	Definición
Transferencia en serie	Serial transfer	Transferencia sucesiva de una serie de elementos de información, de unidad a unidad o de terminal a terminal.
Transferencia radial	Radial transfer	Transferencia de datos entre equipos periféricos y la memoria principal.
Transferir	Transfer	Acción de transmitir uno o varios datos de un punto a otro, escribiéndolos con idéntico contenido en el receptor. Mover información desde unas posiciones de memoria a otras.
Transición	Transition	Cambio, en un circuito, de una condición operativa a otra.
Transitorio	Transient	Perturbación física intermedia que se produce entre dos condiciones correspondientes a estados estables o permanentes.
Transmisión	Transmission	Transferencia eléctrica de una información desde un punto a otro.
Transmisión punto a punto	Point-to-point transmission	Transmisión que se realiza directamente entre dos puntos sin intervención de terminal intermedia ni computadora.
Transmisión síncrona en binario	Binary synchronous transmission	Forma de transmisión en la que el sincronismo de los caracteres se controla por señales de tiempo.
Transmisión-recepción automática	ASR	Terminal que dispone de un perforador de cinta, lector de cinta, teclado e impresora.
Transmisor conectado	X-on	Dentro del protocolo para controlar el flujo de datos entre computadoras u otros dispositivos en conexiones seriales asíncronas la "X" significa "transmisor" y X-on es la señal para iniciar la transmisión.
Transmisor desconectado	X-off	Dentro del protocolo para controlar el flujo de datos entre computadoras u otros dispositivos en conexiones seriales asíncronas la "X" significa "transmisor" y X-off es la señal para terminar la transmisión.
Transmisor receptor asíncrono universal, UART	UART (Universal Asynchronous Receiver Transmitter)	Convertidor serie-paralelo y paralelo-serie.
Transmisor receptor síncrono universal	USRT	Convertidor serie-paralelo para comunicaciones de alta velocidad.
Traza, Segmento	Stroke	Punto o marca utilizado en la formación de caracteres. Elemento del trazado de un carácter impreso.

Término en español	Término en inglés	Definición
Triturador de números	Number cruncher	Término de la jerga que se aplica a las computadoras de gran potencia de cálculo, cuya característica más destacable es su capacidad para manejar cifras grandes a mucha velocidad.
Truncar	Truncate	Acción consistente en suprimir los dígitos de un número que no son significativos, de acuerdo con algún requisito predeterminado en cuanto a la exactitud del resultado.
Tubo de rayos catódicos (TRC), Pantalla de visualización	CRT	Designa el tubo de rayos catódicos instalado en televisores y monitores de computadora. Dentro del tubo, un cañón produce un haz de electrones, que escanea una superficie plana para producir una imagen (obsoleto).
Ubicación, Emplazamiento	Location	Lugar numerado o denominado en memoria en el que una unidad de datos o una instrucción es almacenada.
Ultimo en entrar primero en salir, Pila, (LIFO)	Last-in-first-out (LIFO)	Término que describe un método de extracción de elementos o artículos de una lista ordenada en cola.
Umbral	Threshold	Valor específico establecido para controlar la entrada desde un elemento umbral.
Unidad aritmética y lógica (UAL)	ALU	Parte de la unidad central de proceso que realiza la ejecución de operaciones tanto lógicas como aritméticas.
Unidad central	Main frame	En un principio comprendía la estructura principal de una ***UPC***, en la que estaban instalados la unidad aritmética y la circuitería lógica asociada. Actualmente se refiere, en sentido coloquial, al propio procesador central de gran potencia y tamaño.
Unidad de cinta*	Tape unit	Unidad formada por un transportador de cintas, una cabeza lectora-escritora y los dispositivos correspondientes de control, cuyo conjunto permite tratar las cintas magnéticas en una computadora.
Unidad de control	Control unit	Módulo encargado de buscar y decodificar instrucciones. Esta unidad requiere de un registro de instrucciones, un contador de programa y genera las señales de control y dirige al bus o barra de control.
Unidad de control de periférico	Peripheral control unit	Unidad encargada de controlar las operaciones de una o más unidades periféricas pertenecientes a un sistema de tratamiento automático de datos y que permanecen bajo el control de la ***UPC*** de dicho sistema.

Término en español	Término en inglés	Definición
Unidad de procesamiento central, *UPC*	CPU	Módulo encargado de buscar, decodificar y ejecutar instrucciones. Incorpora una unidad de control, una unidad aritmética y lógica (ALU) así como dispositivos afines (registros, reloj, etc.).
Unidad de proceso	Processing unit	Parte de la computadora que proporciona un área de almacenamiento para programas y datos que efectúa las operaciones especificadas en el programa.
Unidad de salida	Output unit	Dispositivo o dispositivos que se integran en la configuración de un equipo de tratamiento automático de la información, cuya única finalidad es proporcionar la salida al proceso en curso.
Unidad enchufable	Plug-in unit	Conjunto de componentes electrónicos normalizados unidos entre sí por conductores, que puede enchufarse y desenchufarse fácilmente.
Vaciado de memoria	Memory dump	Proceso de transferencia del contenido, total o parcial, de una memoria a otra cualquiera, generalmente para comprobar el contenido de la misma.
Variable	Variable	Cantidad que puede adquirir un valor cualquiera dentro de una serie de valores.
Velocidad de transmisión	Bit rate	Velocidad expresada en bits por segundo. Se abrevia bps.
Velocidad de transmisión	Transmission speed	Número de elementos de información enviados por unidad de tiempo.
Verificación	Check	Cualquier verificación o reconocimiento realizado para evitar la presencia de errores.
Verificación de error	Error Check	Aviso que presenta la máquina al producirse un error.
Verificación de paridad	Parity check	Comprobación que se lleva a cabo durante la transferencia de datos. Consiste en sumar los bits de una unidad de datos, calcular el bit de paridad necesario y comprobar el bit de paridad calculado, comparándolo con el bit de paridad que se transfiere con la partida de datos.
Verificación de validez	Validity check	Verificación o reconocimiento realizado para contrastar la presencia o ausencia de error.
Verificación incorporada	Built-in check	Dícese de determinados dispositivos o técnicas especiales de control y comprobación automática que van incluidos en diversos equipos.
Verificación por redundancia cíclica	Cyclic redundancy check	Método de verificación de errores que se emplea en estación emisora y en la receptora cada vez que se acumula un carácter de verificación de bloques.
Verificar	Verify	Acción de comprobar una transcripción realizada con unos datos. Acción de

Término en español	Término en inglés	Definición
		comprobar, mediante una repetición de la misma operación, si la perforación realizada sobre unas tarjetas o la grabación hecha sobre unas cintas magnéticas por una grabadora, son correctas o no.
Verificar, Comprobar	Check out	Aplicar a un programa un método conducente a un diagnóstico.
Vía de acceso	Port	Parte de un procesador de datos que se dedica a un solo canal de datos con el fin de recibir o transmitir éstos a uno o más dispositivos externos situados a distancia.
Vía, Camino	Path	Secuencia de instrucciones de un programa de computadora.
Visualización en tabla	Table look-at	Localización de un elemento de una tabla por cálculo directo en lugar de efectuar una investigación como en la consulta de tablas.
Volcar	Dump	Operación consistente en copiar el contenido total o parcial de una memoria a otra área o a un listado.
Vuelco estático	Static dump	Vuelco que se realiza cuando un programa llega a un fin de pasada o a alguna otra fase reconocible dentro del proceso.
Vuelco instantáneo	Snapshot dump	Vaciado de partes seleccionadas del almacenamiento que se puede producir en diversos puntos durante el desarrollo de un programa, generalmente para uso en depuración.
Vuelco selectivo	Selective dump	Lectura de un área limitada de memoria.
Y	AND	Operación lógica definida por la regla siguiente: si A y B son verdaderos, C es verdadero; de lo contrario, C es falso.
Yuxtaposición	Juxtaposition	Acción de colocar entidades, elementos o unidades de modo que quedan adyacentes o contiguos.
Zona	Zone	Área que se reserva en memoria para efectuar un trabajo específico.

B Términos en Inglés

Término en inglés	Término en español
A/D	Analógico-digital.
Abort	Interrupción abrupta de un programa o proceso.
Absolute loader	Cargador absoluto.
Access time	Tiempo de acceso.
Accumulator	Acumulador.
Acknowledge	Respuesta afirmativa.
Acoustical coupler	Acoplador acústico.
Adder	Sumador.
Add-on	Ampliación.
Address	Dirección.
Address mark	Marca de dirección.
ALGOL	Lenguaje de programación algorítmico.
Algorithm	Algoritmo.
Algorithmic Language	Lenguaje algorítmico, ALGOL.
Alphanumeric	Alfanumérico.
Alterable memory	Memoria alterable.
ALU	Unidad aritmética y lógica (***UAL***).
Analog/analogue	Analógico.
Analyser/analyzer	Analizador.
AND	"Y".
APL	Lenguaje de programación APL.
ARQ (automatic repetition request)	Petición automática de repetición.
ASCII	Código americano normalizado para el intercambio de información.
ASR (automatic send-receive)	Transmisión-recepción automática.
Assembler	Ensamblador.
Asynchronous	Asíncrono.
Attenuation	Atenuación.
Auto-answer	Respuesta automática.
Automatic calling unit	Dispositivo de llamada automática.
Automatic storage allocation	Asignación de memoria automática.
Auxiliary storage	Almacenamiento auxiliar.
Background program	Programa subordinado.
Backplane	Conexión posterior (entre tarjetas).
Backup copy	Copia de seguridad.
Base register	Registro de base.
BASIC	Código de instrucciones simbólicas de carácter general para principiantes.
Batch	Lote.
Baud	Baudio.
Baudot	Baudot.

Término en inglés	Término en español
BCD (binary coded decimal)	Decimal codificado en binario.
Benchmark program	Programa patrón.
Bi-directional	Bidireccional.
Binary cell	Célula binaria.
Binary number	Número binario.
Binary search	Búsqueda binaria.
Binary synchronous communication	Comunicación binaria síncrona.
Binary synchronous transmission	Transmisión síncrona en binario.
Binary to decimal conversion	Conversión binaria a decimal.
Bipolar	Bipolar.
Bistable	Biestable.
Bit (binary digit)	Bit.
Bit rate	Velocidad de transmisión.
Bits per inch	Bits por pulgada.
Bit-slice microprocessor	Microprocesador expandible.
Blanking	Borrado.
Block	Bloque.
Board	Tarjeta.
Boolean algebra	Algebra de Boole.
Bootstrap	Programa inicializador.
Branch	Bifurcación.
Breakpoint	Punto de interrupción, Punto de ruptura.
Bubble memory	Memoria de burbujas.
Buffer	Memoria intermedia.
Bug	Error.
Built-in check	Verificación incorporada.
Bulk storage	Almacenamiento de gran capacidad.
Bump	Memoria anexa.
Bus	Barra, canal, conductor común, Bus.
Byte	Octeto, byte.
Cache memory	Caché.
Call	Llamada a otra área de memoria.
CAM	Memoria de contenido direccionable.
CAM (computer aided manufacturing)	Manufactura auxiliada por computadora.
Card	Tarjeta.
Card	Tarjeta perforada.
Card file	Archivo de tarjetas.
Carriage	Carro.
Carryover	Acarreo, Arrastre.
Cassette	Casete.
Cerdip	Cerdíp.
Chain	Cadena.
Change record	Registro de modificación.
Channel	Canal.
Channel Analog	Canal analógico.
Channel command	Comando de canal.
Channel duplex	Canal dúplex.

Término en inglés	Término en español
Character	Carácter.
Character alignment	Alineación de caracteres.
Character check	Comprobación de caracteres.
Character error rate	Relación de caracteres erróneos.
Character generator	Generador de caracteres.
Chart	Gráfica, Diagrama.
Check	Verificación.
Check bit	Bit de verificación.
Check out	Comprobación de resultados de salida, Verificar, Comprobar.
Checksum	Suma de verificación.
Clear	Borrar, Restaurar.
Clock	Reloj, Generador de impulsos.
CMOS (complementary metal oxide semiconductor).	***MOS*** complementario.
CMRR	Relación del rechazo en modo común.
COBOL	Lenguaje común para actividades comerciales.
Codec	Codificador-decodificador.
Coded decimal	Decimal codificado.
Combinational logic	Lógica combinacional.
Compiler	Compilador.
Computer	Computador, Ordenador.
Console	Consola.
Control unit	Unidad de control.
Conversational	Interactivo, Conversacional.
Core	Núcleo, Ferrita.
CPS (characters per second)	Caracteres por segundo.
CPU (central processing unit)	Unidad de procesamiento central, ***UPC***.
CPU time	Tiempo de ***UPC***.
CROM (control read only memory)	Memoria de sólo lectura (***ROM***) de control.
Cross program	Programa cruzado.
Crossbar system	Sistema de barras cruzadas.
Crossfoot	Sumar y/o restar horizontalmente.
Crosstalk	Diafonía.
CRT (cathode ray tube)	Tubo de rayos catódicos (TRC), Pantalla de visualización.
Crystal	Cristal.
Current loop	Lazo o bucle de corriente.
Cursor	Cursor.
Custom IC	CI a la medida.
Cutoff	Corte.
CW (control word)	Palabra de control.
Cycle stealing	Robo de ciclos de la ***UPC***.
Cycle time	Tiempo de ciclo.
Cyclic redundancy check	Verificación por redundancia cíclica.
Cyclic shift	Desplazamiento cíclico.
Cyclic store	Almacenamiento cíclico.

Término en inglés	Término en español
D/A (digital/analogic)	Digital/Analógico.
Daisy chain	Cadena en margarita.
Data acquisition	Adquisición de datos.
Data area	Área de datos.
Data bank	Banco de datos.
Data carrier	Portadora, Soporte de datos.
Data cell	Celda de datos.
Data code	Código de datos.
Data collection	Recolección de datos.
Data communication	Comunicación de datos.
Data link escape	Escape de enlace de datos.
DAV (data availability)	Disponibilidad de datos.
Debugging	Depuración, corrección, puesta a punto.
Decade counter	Contador de décadas.
Decay time	Tiempo de declinación.
Deceleration time	Tiempo de deceleración.
Decibel	Decibelio.
Decoder	Decodificador.
Dedicated register	Registro dedicado.
Delay line	Línea de retardo.
Delta noise	Ruido delta.
Demodulation	Demodulación.
Demultiplexer	Demultiplexador.
Development system	Sistema de desarrollo.
Diagnostics	Diagnóstico.
Digitizer	Digitalizador.
Direct access	Acceso directo.
Direct addressing	Direccionamiento directo.
Disable	Neutralización, Deshabilitación.
Diskette*	Diskette, Disco flexible.
DOS	Sistema operativo en disco.
Drum*	Tambor.
Dump	Volcar.
Duplex	Dúplex.
Dynamic memory	Memoria dinámica.
EBCDIC	Código EBCDIC.
Echo	Eco.
Edit code	Código de edición.
Editor	Editor.
EIA-RS232C	Estándar de comunicación serial EIA-RS232C.
Emulation	Emulación.
Enable	Habilitación, Desinhibición.
Encode	Codificar.
End	Final.
Entry	Entrada.
Environment	Configuración.
EPROM (erasable programmable read only memory)	Memoria pasiva (***ROM***) programable.
Erase	Borrar.
Error	Error.

Término en inglés	Término en español
Error Check	Verificación de error.
Error correcting code	Código corrector de errores.
Even parity check	Comprobación paridad par.
Execute cycle	Ciclo de ejecución.
Execution time	Tiempo de ejecución.
Extractor	Extractor.
Fallback	Situación de emergencia.
Fan-in	Carga de entrada.
Fan-out	Carga de salida.
Fetch	Búsqueda.
FIFO (first in first out)	Primero en entrar-primero en salir, Cola de espera, FIFO.
File	Archivo, Fichero.
Firmware	Soporte lógico inalterable.
Flag	Señalizador.
Floating point	Punto flotante.
Floppy disk*	Disco flexible.
Flowchart	Diagrama de flujo, Organigrama.
Flyback	Retorno.
Four wire channel	Canal de cuatro hilos.
Fragmentation	Fragmentación.
Frequency-shift keying (FSK)	Manipulación por desplazamiento de frecuencia.
Full adder	Sumador total, totalizador.
Full duplex circuit	Dúplex total, bidireccional simultáneo.
Gain	Ganancia.
Gangpunch*	Perforación en serie.
Gap	Entrehierro.
Gap	Intervalo, Separación.
Garbage	Información inválida.
Garbage collector	Recolector de Basura.
Gate	Compuerta, puerta.
General purpose computer	Computador para uso general.
Generated address	Dirección generada.
Generator	Generador.
GPIB	Barra de acoplamiento de aplicación general.
Half duplex channel	Canal semidúplex.
Half-adder	Semisuma.
Half-duplex	Semidúplex.
Halt	Parada.
Hamming code	Código Hamming.
Handshaking	Intercambio de señales de control.
Hard-copy	Salida impresa.
Hardware	Electrónica, Componentes, Dispositivos físicos.
Head	Cabezal.
Hexadecimal	Hex, Hexadecimal.
High-level language	Lenguaje de alto nivel.
Highway	Conductor común, Línea principal.

Término en inglés	Término en español
Host computer	Computadora anfitriona, Computadora primaria.
Housekeeping	Preparación previa.
I/O, Input/Output	Entrada/Salida, E/S.
Identify	Identificar.
Idle time	Tiempo pasivo.
IEEE	Instituto de Ingenieros Electrónicos.
Immediate	Inmediato.
Impact printer	Impresora de impacto.
In-circuit emulation	Emulación en circuito.
Increment	Incrementar.
Indexed	Indexado.
Indirect	Indirecto.
Initialization	Inicialización.
Inquiry	Consulta, Interrogación.
In-stream procedure	Procedimiento incorporado.
Instruction	Sentencia única en un programa. Instrucción.
Instruction set	Juego de instrucciones.
Instruction time	Tiempo de instrucción.
Integrated Data Processing (IDP)	Proceso de datos integrados.
Integrated-injection logic (I2L)	Lógica de inyección, Inyección integrada, Lógica I^2L.
Integrator	Integrador.
Interaction	Interacción.
Interface	Acoplamiento, Interconexión.
Interlace	Entrelazar, Concatenar.
Interleaving	Interpolación.
Interpreter	Intérprete.
Interrupt	Interrupción.
Iteration	Iteración.
Job	Trabajo.
Joggle	Empujar, Emparejar tarjetas.
Joystick	Palanca de mando.
Jump	Salto.
Juxtaposition	Yuxtaposición.
K	K. (kilo, 1000 o 10^3).
Key	Llave, Clave, Tecla.
Keyboard	Teclado.
Keyword	Palabra clave.
KIPS	KIPS.
Label	Rótulo, Etiqueta.
Lace	Perforación en cadena.
Lag	Retardo.
Language	Lenguaje.
Language character set	Juego de caracteres del lenguaje.
Language Object	Lenguaje objeto, Lenguaje máquina.
Language translator	Traductor de lenguaje.
Last-in-first-out (LIFO)	Ultimo en entrar primero en salir, Pila, (LIFO).
Latch	Cerrojo.

Término en inglés	Término en español
Latency	Espera.
Learning curve	Curva de aprendizaje.
Least significant bit (LSB)	Bit menos significativo.
Library	Biblioteca.
Line delay	Línea de retardo.
Line printer	Impresora de líneas.
Link	Enlace.
Linkage	Enlace.
Listing	Listado.
Loader	Cargador.
Location	Ubicación, Emplazamiento.
Lockout	Bloqueo.
Logger	Registrador.
Log-on	Identificarse.
Look-up table	Tabla de consulta.
Loop	Bucle, Lazo.
Low order	Orden inferior.
LPM (Lines per minute)	Líneas por minuto.
Machine	Máquina.
Machine code	Código de máquina.
Machine error	Error de máquina.
Machine instruction	Instrucción de máquina.
Machine language	Lenguaje de máquina.
Macro instruction	Macroinstrucción.
Magnetic core storage	Almacenamiento de núcleo magnético.
Magnetic disk	Disco magnético.
Main frame	Unidad central.
Main memory	Memoria principal.
Main program	Programa principal.
Main storage	Almacenamiento principal.
Mark	Marca.
Masking	Enmascaramiento, Mascarilla.
Mass storage	Almacenamiento masivo.
Matching	Comparación con selección.
Matrix store	Memoria matricial.
Mega	Mega.
Memory	Memoria.
Memory dump	Vaciado de memoria.
Memory fill	Carga de memoria.
Memory guard	Protección de memoria.
Mercury memory*	Memoria de mercurio.
Merge	Compaginado, Fusión.
Micro	Micro.
Microinstruction	Microinstrucción.
Milli	Mili.
Misfeed	Pérdida de alimentación.
Mistake	Error, Equivocación.
Modem	Modulador/demodulador, Modem.
Monitor	Monitor.
Most-significant-character (MSC)	Carácter más significativo.

Término en inglés	Término en español
Multiprocessor	Multiprocesador.
Multi-access	Acceso múltiple, Multiacceso.
Multiple address	Dirección múltiple.
Multiplexor	Multiplexor.
Multiprocessing	Multiproceso.
Multiprogrammation	Multiprogramación.
Multisequential system	Sistema multisecuencial.
Multistation	Circuito multiterminal.
N+one address instruction	Instrucción de n+una dirección.
NAND	NO-Y.
Nanosecond	Nanosegundo.
Natural function generator	Generador de función natural
Natural language	Lenguaje natural.
N-core per-bit store	Memoria de N núcleos por bit.
Nesting	Jerarquización, Anidamiento.
Network	Red.
No-address instruction	Instrucción sin dirección.
Noise	Ruido.
Non-return-to-zero recording	Grabación sin retorno a cero.
Non-volatile memory	Memoria no volátil, Memoria estable.
NOR-gate	Compuerta NO-O.
NOT-AND	NO-Y.
Null instruction	Instrucción nula.
Number cruncher	Triturador de números.
Numeric character	Carácter numérico.
Numeric coding	Codificación numérica.
Numeric word	Palabra numérica.
Object code	Código objeto.
Object configuration	Configuración objeto.
Object language	Lenguaje objeto.
Object program	Programa objeto.
Object routine	Rutina objeto.
OCR (optical character recognition)	Reconocimiento de caracteres ópticos.
Octal	Octal.
Octet	Octeto.
Odd-even check	Control de paridad par-impar.
Odd-parity check	Comprobación de paridad impar.
Off-line	Fuera de línea, Autónomo.
Off-line unit	Dispositivo autónomo.
Offset	Desfase.
Off-time	Tiempo de inactividad.
On-line	En línea, en conexión directa.
Open loop	Bucle abierto.
Operand	Operando.
Operating system	Sistema operativo.
Operation	Operación.
Operation register	Registro de operaciones.
Optical-bar code reader	Lector óptico de códigos de barras.
Ordering	Ordenación.

Término en inglés	Término en español
OR-gate	Compuerta tipo "O".
Outboard recorder	Grabación externa.
Output	Salida.
Output queue	Cola de salida.
Output stream	Corriente de salida.
Output unit	Unidad de salida.
Overflow	Rebasamiento.
Overlapping	Solapamiento.
Overlay	Recubrimiento, Traslape.
Overload	Sobrecarga.
Overwrite	Sobrescribir, Sobregrabar.
Pack	Lote, Archivo de tarjetas.
Padding	Relleno.
Page-at-a-time printer	Impresora de páginas completas.
Paper tape*	Cinta de papel perforado.
Paper throw*	Salto de papel.
Parallel	Paralelo, en paralelo.
Parallel storage	Almacenamiento en paralelo.
Parallel transfer	Transferencia en paralelo.
Parameter	Parámetro.
Parity bit	Bit de paridad.
Parity check	Verificación de paridad.
Pass	Pasada.
Patch	Parche.
Path	Vía, Camino.
Pen Light	Lápiz fotosensible.
Peripheral control unit	Unidad de control de periférico.
Peripheral processor	Procesador periférico.
Peripheral unit	Periférico.
Picosecond	Picosegundo.
Pinboard	Tablero de conexiones.
PL/I	Lenguaje PL/I.
Plotter	Graficador, Trazador de gráficos.
Plug-in unit	Unidad enchufable.
Point	Punto.
Pointer	Indicador.
Point-to-point transmission	Transmisión punto a punto.
Polling	Sondeo.
Port	Vía de acceso.
Pre-edit	Preeditar.
Printout	Impresión de salida.
Problem-oriented language	Lenguaje orientado a problemas.
Process	Proceso.
Processing unit	Unidad de proceso.
Processor	Procesador, Unidad de proceso.
Program testing time	Tiempo de prueba del programa.
Program/programme	Programa.
Programming	Programar, Programación.
Propagation delay	Retardo de propagación.
Pseudo-instruction	Pseudo instrucción, Instrucción falsa.

Término en inglés	Término en español
Punch*	Perforar.
Push down store	Memoria de almacenamiento descendente.
Quantizer	Cuantificador.
Quantum	Cuanto.
Queue	Cola de espera.
Quotient	Cociente.
Radial transfer	Transferencia radial.
Radix	Raíz, Base.
Radix complement	Complemento de la base.
Random-access memory (RAM)	Memoria de acceso aleatorio, Memoria ***RAM***.
Random number sequence	Secuencia de números aleatorios.
Range	Rango, Escala.
Rank	Jerarquizar.
Raw data	Datos en bruto.
Read	Leer.
Read time	Tiempo de lectura.
Read-only storage (ROM)	Memoria de sólo lectura, ***ROM***, Memoria pasiva, Memoria inalterable.
Read-out	Lectura de salida.
Real-time	Tiempo real.
Record	Registro.
Redundancy	Redundancia.
Redundant code	Código de redundancia.
Reel	Rollo, Bobina.
Reference time	Tiempo de referencia.
Register	Registro.
Relative address	Dirección relativa.
Relative code	Código relativo.
Relative coding	Codificación relativa.
Relative error	Error relativo.
Relocate	Reubicar.
Remainder	Resto.
Repertoire/repertory	Repertorio, Juego.
Report	Informe.
Rerun point	Punto de recuperación.
Reset	Restaurar poner a cero.
Reset pulse	Pulso de inicialización.
Resident routine	Rutina residente.
Residual error	Error residual.
Response duration	Duración de respuesta.
Response time	Tiempo de respuesta.
Restart	Reanudar.
Restore	Restaurar.
Retrieval	Recuperación.
Rewind*	Rebobinar.
Rewrite	Regrabación, Reescritura.
Rise time	Tiempo de subida.
Rolling	Reincorporar a memoria.
Rollback	Repetir.

Término en inglés	Término en español
Roll-out	Descargar a la memoria externa.
Round	Redondear.
Rounding-off	Redondeo.
Routine	Rutina.
Routing	Encaminamiento, Encauzamiento.
Row	Fila.
Row pitch	Paso entre filas, Paso.
Run	Ejecución de un proceso, Pasada de máquina.
Run phase	Fase de ejecución.
Run time	Tiempo de ejecución, Tiempo de proceso.
Sample and hold	Muestreo y retención.
Sampling	Muestreo.
Scale	Cambio de escala.
Scale	Escala.
Scan	Explorar.
Scanning rate	Secuencia de exploración.
Scatter-read	Lectura dispersa.
Scheduled maintenance	Mantenimiento previsto o programado.
Scientific language	Lenguaje científico.
Scratch pad memory	Memoria de trabajo.
Scratch tape*	Cinta reutilizable.
Search	Búsqueda.
Search time	Tiempo de búsqueda.
Seek	Búsqueda, Posicionar.
Segment	Segmentar, Segmento.
Select	Seleccionar.
Selective dump	Vuelco selectivo.
Selector	Selector.
Self-checking code	Código de autoverificación.
Self-resetting loop	Bucle restaurador.
Self-triggering program	Programa auto inicializado.
Semantics	Semántica.
Sense	Leer, Captar, Detectar.
Sentinel	Centinela, Señalizador.
Sequence	Secuencia.
Sequential access storage	Almacenamiento de acceso secuencial.
Sequential control	Control secuencial.
Sequential processing	Proceso secuencial.
Sequential stacked job control	Control de trabajos agrupados en secuencia.
Serial	Serie.
Serial number	Número de serie.
Serial transfer	Transferencia en serie.
Serviceability	Índice de utilidad.
Set	Conjunto, Juego.
Setup	Preparación, Puesta a punto.
Shift	Desplazar.
Shift out	Desplazar bits uno a uno hacia la salida.

Término en inglés	Término en español
Shift register	Registro de desplazamiento.
Significant digits	Dígitos significativos.
Simplex	Simplex, Unidireccional.
Simulator	Simulador.
Single-ended amplifier	Amplificador de un solo extremo.
Skip	Saltar.
Slice	Limitar.
Slicer	Circuito amplificador de impulsos. Limitador.
Smooth	Ajustar, Nivelar, Filtrar.
Snapshot dump	Vuelco instantáneo.
Software	Programa, Soporte lógico, Conjunto de programas, software.
Sort	Clasificar, Ordenar.
Source code	Código fuente.
Source language	Lenguaje fuente.
Stack	Pila.
Stand-alone	Autónomo, Independiente.
Standby	Espera, En espera.
Standby computer	Computadora de reserva.
State table	Tabla de estados, Tabla de transición de estado.
Statement	Instrucción en lenguaje fuente. Sentencia.
Static dump	Vuelco estático.
Static memory	Memoria estática.
Status	Estado.
Status register	Registro de estado.
Step	Ejecutar una instrucción de un programa.
Stop bit	Bit de parada.
Stop signal	Señal de parada.
Storage	Almacenamiento. Memoria.
Storage cell	Celda de almacenamiento.
Stored program	Programa almacenado.
Straight-line coding	Codificación rectilínea.
Stream	Flujo. Corriente.
String	Cadena.
String length	Longitud de la serie.
Strobe	Muestreo.
Stroke	Traza, Segmento.
Stunt box	Supresor de impresión, Caja reguladora.
Style	Estilo.
Subroutine	Subrutina.
Subset	Subconjunto.
Subtraction	Sustracción.
Sum	Suma.
Summer	Sumador, Sumador analógico.
Support programs	Programas de apoyo.
Symbol	Símbolo.
Symbolic coding	Codificación simbólica.
Synchronization	Carácter de sincronización.

Término en inglés	Término en español
character	
Synchronizer	Sincronizador.
Syntax	Sintaxis.
Table	Tabla.
Table look-at	Visualización en tabla.
Table look-up	Consulta de tablas, Búsqueda de tablas.
Tabulate	Tabular.
Tag	Rótulo, Distintivo.
Tag format	Formato de la etiqueta.
Takedown	Desmontaje.
Tally	Cuenta, Recuento.
Tape*	Cinta.
Tape drive*	Impulsor de cinta.
Tape file*	Archivo de cinta.
Tape reader*	Lector de cinta.
Tape unit*	Unidad de cinta.
Tape wound core*	Núcleo de bobinado magnético.
Target language	Lenguaje resultante, Lenguaje objeto.
Target phase	Fase objeto.
Target program	Programa resultante.
Task	Tarea.
Teleprocessing	Teleproceso.
Teletype*	Teletipo (marca registrada).
Teletypewriter*	Telescritor.
Telex*	Télex.
Temporary storage	Almacenamiento temporal.
Tens complement	Complemento a diez.
Terminal	Terminal.
Terminal repeater	Repetidor terminal.
Terminate	Terminar, Concluir, Finalizar, Suspender, Interrumpir.
Test	Probar, Examinar.
Test program	Programa de verificación.
Test routine	Rutina de verificación.
Test run	Corrida de prueba, Pasada de prueba.
Text	Texto.
Text buffer	Memoria intermedia de textos.
Three-input adder	Sumador de tres entradas.
Three-input subtractor	Restador de tres entradas.
Three-plus-one address	Dirección de tres más uno.
Threshold	Umbral.
Threshold element	Elemento umbral.
Throughput	Rendimiento específico, Productividad.
Time access	Tiempo de acceso.
Time slicing	División o reparto del tiempo
Time-division multiplexing	Multiplexado por división de tiempo.
Time-share	Tiempo compartido.
Timing	Sincronización, Temporización.
Trace	Rastreo, Diagnóstico, Trazado.
Track	Pista.

Término en inglés	Término en español
Track pitch	Paso entre pistas. Separación entre pistas.
Transfer	Transferir.
Transfer function	Función de transferencia.
Transfer instruction	Instrucción de transferencia.
Transfer time	Tiempo de transferencia.
Transient	Transitorio.
Transition	Transición.
Translator	Traductor.
Transmission	Transmisión.
Transmission loss	Pérdida de transmisión.
Transmission speed	Velocidad de transmisión.
Trap	Trampa, Desvío.
Troubleshooting	Localización de errores, Investigación de averías.
Truncate	Truncar.
Truth table	Tabla de decisión lógica. Tabla de verdad.
Turnaround time	Tiempo de respuesta.
Two wire channels	Canal de dos hilos, Par.
Two-address instruction	Instrucción de dos direcciones.
Two-core-per-bit store	Memoria de dos núcleos por bit.
Two-input adder	Sumador de dos entradas.
Two-input subtractor	Restador de dos entradas.
Two-level subroutine	Subrutina de dos niveles.
Two-out-of-five code	Código de dos a cinco.
Two's complement	Complemento a dos.
Type bar	Barra de tipos.
Type wheel	Rueda de tipos.
UART (Universal Asynchronous Receiver Transmitter)	Transmisor receptor asíncrono universal, UART.
Unary operation	Operación unitaria.
Unattended operation	Funcionamiento sin operador.
Unattended station	Estación sin operador.
Unconditional jump	Salto incondicional.
Unitary code	Código unitario.
Unpack	Desempaquetar, Desagrupar.
Unused time	Tiempo de no utilización.
Update	Actualizar.
Uptime	Tiempo productivo, Tiempo activo.
USRT (universal synchronous receptor and transmitter)	Transmisor receptor síncrono universal.
Utility program	Programa de utilidad.
Utility routine	Rutina de utilidad.
Validity check	Verificación de validez.
Variable	Variable.
Variable block	Bloque variable.
Variable-length instruction	Instrucción de longitud variable.
Verify	Verificar.

Término en inglés	Término en español
Virtual address	Dirección virtual.
Voice grade channel	Canal acústico.
Volatile storage	Memoria volátil.
Waiting queue channel	Cola de espera en canal.
Waiting time	Tiempo de espera.
Walk down	Pérdidas acumuladas.
Waste instruction	Instrucción no operativa.
Willful intercept	Interceptación premeditada.
Word	Palabra.
Word length	Longitud de palabra.
Word mark	Marca de palabra.
Word time	Tiempo de palabra.
Work area	Área de trabajo.
Write	Escribir, Grabar.
Write pulse	Impulso de escritura.
Write time	Tiempo de escritura.
Write up	Documentación de programa.
X punch*	Perforación X.
Xerographic printer	Impresora xerográfica.
X-off	Transmisor desconectado.
X-on	Transmisor conectado.
Xtal	Cristal.
X-Y plotter	Graficador, Trazador de gráficos X-Y.
Y-punch*	Perforación Y.
Zero	Cero, Nada.
Zero access storage	Memoria de tiempo de acceso cero.
Zero address instruction format	Formato de instrucción sin dirección.
Zero fill	Rellenar con ceros.
Zero flag	Señalizador de cero.
Zero output signal	Señal de salida cero.
Zeroize	Poner a ceros.
Zone	Zona.
Zone bits	Bits de zona.
Zone punch*	Perforación de zona.
* **Término, definición o concepto obsoleto o en desuso.**	

C Algunas Abreviaciones Comunes

Abreviación	Significado
µs	Microsegundos.
µV	Microvoltios.
AC	Corriente alterna (Alternating current).
adj-ch set	Selectividad de canal adyacente.
alt-ch set	Selectividad de canal alterno.
AM	Amplitud modulada
amp	Amperio, Amper.
ANL	Limitador automático de ruido
ANSI	Instituto americano nacional de estándares (American National Standards Institute).
aux	Auxiliar.
avg	Promedio.
Bluetooth	Anglicismo del nombre de un rey danés. Conjunto de protocolos de transmisión inalámbrica.
BTL	Balanceo sin transformador.
BTL/COM	Balanceo sin transformador (de entrada) con tierra común.
BTL/OCL	Balanceo sin transformador (de entrada) con capacitores de salida.
CA	Corriente alterna.
cap	Captura.
CCA	Amperes de arranque en frío (Cold cranking ampers).
cd	Corriente directa.
ch	Canal (channel).
CI	Circuito Integrado.
cont	Continuo.
CRT	Cathod Ray Tube Pantalla de rayos catódicos
cu	Cúbicos (cubic).
D/A	Digital a analógico.
dB	Decibel.
dBA	Decibeles (medido en Amperios).
dBf	Decibeles por ***FEM***/Watts.
dBV	Decibeles referidos a 1 Voltio.
dc	Corriente directa. (Direct current).
DIMM	Dual in-line memory module. Memoria dispuesta en línea (contiene circuitos DRAM).
DIN	Norma Industrial Alemana (Deutsche Industrie Normen).
DNR	Reducción dinámica de ruido.
DRAM	Dynamic read–write random access memory. Memoria dinámica de lectura/escritura con acceso aleatorio.
DSP	Procesamiento digital de señales.
DTMF	Modulación en frecuencia de dos tonos (Dual Tone Modulated frequency).

Abreviación	Significado
DX	Distante.
ECC	Error-correcting code. Código con corrección de error.
EIA	Asociación de industrias electrónicas (Electronics Industries Association).
EMI	Interferencia electromecánica.
EQ	Ecualizador, ecualización.
FAT	File Allocation Table. Tabla de asignación de Archivos
FET	Transistor de efecto de campo.
Firmware	Programa grabado en ***PROM***.
FM	Frecuencia modulada.
FPU	Floating Point Unit. Unidad de punto flotante.
FR	Respuesta en frecuencia.
FS	Resonancia en el aire (Free-air resonance).
ft	Pies.
H	Horas.
Hardware	Conjunto de circuitos electrónicos, ferretería y periféricos que forman una computadora.
HEXFET	FET hexagonal.
HTML	Hypertext Makup Language. Lenguaje de marcas de hipertexto.
HTTP	Hypertext Transfer Protocol. Protocolo de transferencia de hipertexto.
Hz	Hertz.
IC	Circuito integrado (Integrated circuit).
IDE	Integrated Device Electronics. Interfaz estándar para disco duro.
IEC	Comisión electrotécnica internacional (International Electrotechnical Commission).
IHF	Instituto de alta fidelidad (Institute of High Fidelity).
IM	Distorsión de intermodulación.
imp	Impedancia.
in	Pulgadas (inches).
IP	Protocolo de Internet.
ISO	Organización de estándares internacionales (International Standards Organization).
ISP	Internet Service Provider. Proveedor de servicios de Internet.
k	Kilo (10^3) o karat.
Kernel	Núcleo. El centro del sistema operativo.
kHz	Kilohertz.
L/R	Derecha/izquierda.
LAN	Local Area Network. Red de area local.
lb	Libras.
LCD	Despliegue de Cristal de cuarzo líquido (Liquid Cristal Display).
LED	Diodo emisor de luz (Light emitting diode).
mA	Miliampers.
máx.	Máximo.
MDF	Panel de fibra de vidrio de mediana densidad

Abreviación	Significado
	(Medium /density fiberboard).
MHz	Megahertz.
min	Mínimo.
MOSFET	FET semiconductor de metal-óxido.
ms	milisegundos
mV	Milivoltios(10^{-3}v)
NAM	Modulo asignado por número.
nom	Nominal.
NR	Reducción de ruido.
oct	Octava.
OEM	Fabricante de equipo original (Original equipment manufacturer).
OFC	Cobre libre de oxígeno.
oz	Onzas.
PDM	Modulación por densidad de pulsos (Pulse code modulation).
PIC	Peripheral Interrupt Controller. Multiplexa las interrupciones presentándolas al ***CPU***.
PIC	Programmable Interface Controller. Circuito completo que incluye ***CPU***, memoria e interfaz y es programable.
PLL	Lazo de seguimiento en fase (Phase-locked loop).
PWM	Modulación por amplitud de pulsos (Pulse-width Modulation).
RAID	Redundant array of independent disks. Configuración de discos duros redundantes que reducen el impacto de pérdida de datos.
RAM	Memoria de acceso aleatorio (Random access memory).
RF	Radio frecuencia.
RFI	Interferencia de radio frecuencia.
RISC	Reduced instruction set computer. Computadora que trabaja con un conjunto reducido de instrucciones.
rms	Raíz cuadrada media.
ROM	Memoria de sólo lectura (Read only memory).
RTA	Analizador en tiempo real (Real time analyzer).
S/N	Relación señal a ruido (Signal to noise ratio).
SCR	Silicon-Controlled Rectifier Diodo de cuatro capas
sec	Segundos.
sens	Sensibilidad.
sep	Separación.
SIMM	Single in-line memory module. Módulo de memoria limitado a 64 Mbyte.
SPL/W/m	Nivel de presión de sonido con una entrada de 1 Watt medida a 1 metro.
SRAM	Static ***RAM***. Memoria ***RAM*** estática.
SSR	Solid State Relay. Relé de estado sólido
THD	Distorsión armónica total.
TIM	Distorsión temporal de intermodulación.
TRIAC	Triod for Alterning Current Switch usado en la construcción de relés de estado sólido

Abreviación	Significado
V	Voltios, Volts.
VLSI	Very large-scale integrated circuit. Técnica de fabricación de circuitos de integración a gran escala.
W	Watts, vatios.
WiFi	Wireless Fidelity. Conjunto de protocolos de transmisión inalámbrica.
w/&F	Distorsión de audio (Wow and flutter).
wrms	Raíz media cuadrática balanceada (Weighted root medium square).

D MS-Debug

Bug (término en inglés) significa insecto, pero en informática se usa en el sentido de fallo o defecto en un programa. Históricamente queda implantado en su acepción en el sentido de depurar (escrutar y eliminar fallos). Se usa tanto como sustantivo o verbo, de la que han derivado otros. Por ejemplo: Debugger (depurador). Por extensión, todos los programas y utilidades que sirven para escudriñar los datos y el código a bajo nivel, se identifican genéricamente con esta denominación.

DEBUG es una utilidad externa que los fabricantes del sistema operativo (*SO*) de MS-DOS (Microsoft® y sus competidores o de una alternativa libre como Linux®) incluyen y que permite visualizar la memoria, introducir pequeños programas y rastrear su ejecución.

Con la llegada de *SO* visuales basados en ventanas (Windows®) su uso se basa en la interfaz de comandos y, siendo una aplicación de 16 bits, se limita a aquellos *SO* de 32 bits. En los SO de 64 bits no se incluye, pero existen alternativas:

- Instalar una máquina virtual con un SO adecuado de 32 bits.
- Usar un emulador de MS-DOS® para su versión especifica de Windows.
- Usar el depurador que ofrecen gratuitamente otras compañías.
- Usar el depurador de Linux® (que permite practicar y aprender las bases del ensamblador).

Una de las características importantes del programa *DEBUG* es que despliega todo el código del programa en formato hexadecimal.

Es importante saber que muchos usos de este tipo de utilidades de bajo nivel, requieren un funcionamiento independiente o desligado de un Sistema Operativo multi-usuario, ya que éstos encapsulan y ocultan muchos aspectos del hardware a los que este tipo de programa tienen acceso directo.

DEBUG es útil, entre otras cosas, para:

DEBUG

Tradicionalmente, todas las computadoras y sistemas operativos han incluido una función de mantenimiento, que se utiliza para determinar si un programa funciona correctamente o no. *DEBUG* fue escrito originalmente por Tim Paterson para cumplir este propósito en 86-DOS. Cuando Paterson comenzó a trabajar para Microsoft a principios de la década de los 80, usó esta herramienta como parte de DOS v1.00 y desde entonces se ha incluido en MS-DOS/PC DOS y en ciertas versiones de Microsoft Windows.

- Propósitos educativos.
- Ensamblar pocas líneas de código.
- Desensamblar código en RAM, ROM y ejecutables.
- Ejecutar paso a paso programas.
- Desplegar datos en memoria.
- Verificar el estado de los registros del CPU.
- Eliminar virus.

D.1 Comandos

En la tabla *D.1* se resumen los comandos que puede utilizar en *DEBUG*.

Tabla D.1	
Comandos de MS-Debug	
Instrucción	**Comando y Parámetros**
Ayuda	?
Ensamblar	A [dirección]
Comparar	C Rango de direcciones
Volcar	D[B|W|D] [rango]
Volcar Interrupción	DI Interrupción [cuenta]
Volcar LDT	DL Selector [cuenta]
Volcar cadena MCB	DM
Volcar memoria externa	DX
Editar	E dirección [lista de valores]
Rellenar	F lista de intervalos
Ir a	G [=dirección] [puntos de interrupción]
Hex	H valor1 valor2
Entrada	I[W|D] puerto
Cargar archivo	L [dirección] [unidad] [sector] [cuenta]
Mover	M rango de direcciones separados por comas
Modo x86	M [x] [x=0…6]
Modo FPU	MC [2|N] (2=287,N=no FPU)
Nombrar	N [[disco:][ruta]nombre_archivo [lista_argumentos]]
Salida	O [W|D] puerto valor
Continuar	P [=dirección] [cuenta]
Salir	Q
Registro	R [registro]
Registro MMX	RM
Registro FPU	RN[R]
Intercambia con registros 386	RX
Buscar	S lista de rangos
Rastrear	T [=dirección] [valor]

Modo de Rastreo	TM [0|1]
Desensamblar	U [rango]
Guardar	W [dirección] [disco] [sector] [cuenta]
Asignar memoria expandida	XA [#páginas]
Desasignar memoria expandida	XD [identificador]
Asignar páginas de memoria expandida	XM [Lpágina] [Ppágina] [identificador]
Mostrar estado de la memoria expandida	XS
Nota: Los parámetros entre corchetes ([]) son opcionales. Los parámetros opcionales suelen indicar que hay varias formas diferentes de utilizar un comando.	

D.1.1 Parámetros

Podrá notar que la mayoría de las instrucciones aceptan uno o varios parámetros

D.1.1.1 Dirección

Ubicación de memoria especificada en hexadecimal. Puede usar un Desplazamiento simple por sí mismo (en cuyo caso, se asumirá el CS[66] actual), o puede ingresar el Segmento:Desplazamiento completo utilizando números hexadecimales o sustituyendo el nombre de un registro de segmento por un número. No se requieren ceros iniciales; por lo tanto, 1F sería CS:001F (CS significaría el valor que CS tuviese al momento de teclear esto). Ejemplo:

```
100 DS:12 SS:0 198A:1234
```

D.1.1.2 Rango

Dos direcciones hexadecimales separadas por un solo espacio. Se pueden enumerar como pares completos de Segmento:Desplazamientos o un Desplazamiento único (en cuyo caso el segmento se asume como el valor presente del CS actual). Nota: Algunos comandos, tal como Comparar (C), pueden requerir que se proporcione una segunda dirección como un desplazamiento.

---

[66] Segmento de código.

D.1.1.3 Lista

Una cadena de bytes hexadecimales separados por un espacio o datos ASCII encerrados entre comillas simples o dobles. Puede enumerar cualquier número de bytes desde uno hasta el número que quepa en la línea antes de tener que presionar la tecla *Intro*. Un solo byte, tal como 00, se usa frecuentemente con el comando *FILL* (f) mientras que el comando *ENTER* (e) probablemente tendrá una cadena de muchos bytes hexadecimales o caracteres *ASCII* por línea; por ejemplo:

```
e 100 31 C0 B4 09 BA 50 02 CD 21 B8 4C 00 CD 21
e 250 'Esta es una cadena de datos ASCII.$'
```

D.1.1.4 Número

Acuérdese que todos los números y valores usados en cualquier comando *DEBUG* se entienden como hexadecimales solamente. Eso incluye el número de sectores en los comandos *LOAD* o *WRITE* e incluso el número de instrucciones que desea que *DEBUG* recorra en los comandos *TRACE* o *PROCEED*. Y recuerde:

¡Todo es en hexadecimal todo el tiempo dentro del *DEBUG*!

D.2 Tutorial

Presentamos en esta sección una serie de sencillos ejemplos que le darán la pauta a seguir para armar otros más complicados conforme avanza en su comprensión del lenguaje ensamblador de la familia x86.

Nota importante: todos los ejemplos han sido ejecutados en una máquina virtual con un sistema operativo Windows 10 PRO de 32 bits instalado, por lo que el resultado de ciertos comandos pueden diferir de lo que usted obtenga al ejecutarlos.

D.2.1 Ejecutando DEBUG

DEBUG es una utilería que funciona a partir de la línea de comandos y, por lo tanto, debe ejecutarse a partir de una ventana *DOS*. Para ello se usa el comando *CMD* a partir de la aplicación Windows. Verá abrirse la siguiente ventana:

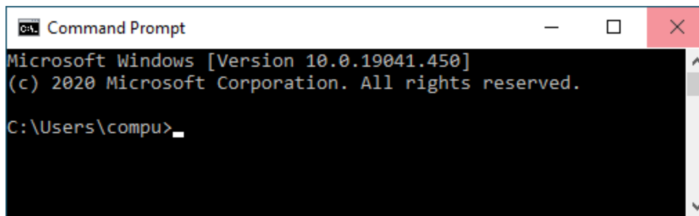


Figura D.174 Ventana de línea de comandos.

Es a partir de esta ventana que se ejecutan todos los comandos del *SO*, entre ellos *DEBUG*.

Una vez abierta la ventana, escriba *DEBUG* (mayúsculas o minúsculas) y presione la tecla Intro para aceptar la entrada y ejecutar el programa. La siguiente pantalla aparecerá:

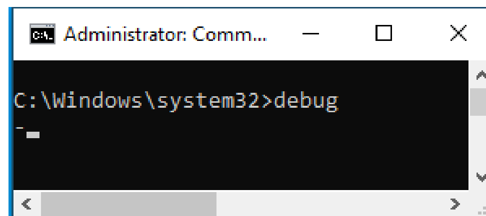


Figura D.175 Ejecutando DEBUG.

¡No muy informativa!

En las siguientes secciones analizaremos los comandos mas importantes de este programa y daremos pequeños ejemplos de su uso.

D.2.2 Detalles de los Comandos más Importantes

D.2.2.1 Ayuda: ?

`Parámetro: No tiene`

El signo de interrogación sirve para solicitar ayuda y es de los primeros comandos a ejecutar y aprender junto con salir del *DEBUG* (*Q*). Vea la siguiente figura:

```
Administrator: Command Prompt
C:\Windows\system32>debug
-?
assemble     A [address]
compare      C range address
dump         D [range]
enter        E address [list]
fill         F range list
go           G [=address] [addresses]
hex          H value1 value2
input        I port
load         L [address] [drive] [firstsector] [number]
move         M range address
name         N [pathname] [arglist]
output       O port byte
proceed      P [=address] [number]
quit         Q
register     R [register]
search       S range list
trace        T [=address] [value]
unassemble   U [range]
write        W [address] [drive] [firstsector] [number]
allocate expanded memory        XA [#pages]
deallocate expanded memory      XD [handle]
map expanded memory pages       XM [Lpage] [Ppage] [handle]
display expanded memory status  XS
-q
```

Figura D.176 Solicitar ayuda.

D.2.2.2 Salir: Q

`Parámetro: No tiene`

Con este comando se sale inmediatamente del programa DEBUG. No se hacen preguntas. Debe ser el primer comando que recuerde junto con el de Ayuda "?".

D.2.2.3 Ejecutar un Programa

`Parámetro: [[disco:][camino]nombre [argumentos]]`

Ejecuta DEBUG junto con un programa previamente editado en forma de secuencia de comandos.

Sus parámetros son:

`[[disco:][camino]nombre`

Especifica el archivo a probar, por ejemplo:

`d:\myejemplo\pantalla.asm`

y

`[argumentos]`

Especifica los argumentos o parámetros requeridos por su programa (si es que se requieren).

D.2.2.4 Hex: H

```
Parámetro: valor1 valor2
```

Forma una especie de calculadora hexadecimal muy simple (sólo sumar y restar). No olvide que todos los números dentro de *DEBUG* son siempre hexadecimales. Escriba dos valores hexadecimales (de no más de cuatro dígitos cada uno) y el programa le mostrará primero la *SUMA*, luego la *DIFERENCIA* de esos valores.

Ejemplos:

```
-h aaa 531
0FDB 0579

-h fff 3
1002 0FFC
```

D.2.2.5 Volcar: D

```
Parámetro: [dirección] [longitud]
```

Muestra el contenido de un bloque de memoria. Si se usa sin parámetros, el comando le mostrará los primeros 128 bytes a partir de la posición a la que se llegó en el último comando D utilizado.

Las ubicaciones de la memoria cerca del inicio del segmento C000 le mostrarán información sobre el tipo de tarjeta de vídeo instalada en su PC. El siguiente ejemplo le informa que el sistema emula una tarjeta IBM EGA:

```
-d c000:0010
C000:0010  20 45 6D 75 6C 61 74 69-6F 6E 20 6F 66 20 49 42    Emulation of IB
C000:0020  4D 20 45 47 41 20 52 4F-4D 20 56 69 64 65 6F 20   M EGA ROM Video
C000:0030  53 65 76 65 6E 20 42 49-4F 53 20 43 6F 64 65 2C   Seven BIOS Code,
C000:0040  20 56 65 72 73 69 6F 6E-20 31 2E 30 34 20 00 00    Version 1.04 ..
C000:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 56 37   ..............V7
C000:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
C000:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
C000:0080  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-
```

D.2.2.6 Buscar: S

```
Parámetro: rango
```

Busca dentro de un rango de direcciones un patrón de uno o más valores de bytes dados en una lista. La lista puede estar formada por números o cadenas de caracteres entre comillas simples o dobles. Ejemplo:

```
-s 0000:0 ffff "BIOS"
0000:0C7C
0000:A237
0000:AEA7
0000:AEB6
-d 0:0c7c
0000:0C70                                      42 49 4F 53               BIOS
0000:0C80  22 0D 0D 00 00 00 00 00-00 00 00 00 00 00 00 00  "...............
0000:0C90  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
0000:0CA0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
0000:0CB0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
0000:0CC0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
0000:0CD0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
0000:0CE0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
0000:0CF0  00 00 00 00 00 00 00 00-00 00 00 00              ............
-
```

D.2.2.7 Comparar: C

```
Parámetro:  rango de direcciones
```

Coteja dos bloques de memoria. Si no hay diferencias, *DEBUG* simplemente muestra otro mensaje (-). A continuación, mostramos un ejemplo con diferencias:

```
-c 140 148 340
0B1B:0140  AA  32  0B1B:0340
0B1B:0141  81  ED  0B1B:0341
0B1B:0142  CD  0B  0B1B:0342
0B1B:0143  00  C9  0B1B:0343
0B1B:0144  40  74  0B1B:0344
0B1B:0145  EB  0D  0B1B:0345
0B1B:0146  CE  8D  0B1B:0346
0B1B:0147  E8  6F  0B1B:0347
0B1B:0148  2B  09  0B1B:0348
```

El comando anterior solicita que las localidades 140 hasta la 148 se comparen a la 340 (se implica que hasta la 348); los bytes diferentes se despliegan lado a lado (con su localización exacta, incluyendo el segmento en ambos lados).

D.2.2.8 Rellenar: F

```
Parámetro: Lista de rangos
```

Este comando puede usarse para:

- Llenar una zona de memoria con un valor determinado.
- Borrar grandes áreas de la memoria.
- Llenar áreas más pequeñas con una frase que se repite continuamente o un solo byte.
- Como, al terminar la ejecución de un programa, la zona de memoria en que se residía no se borra (poniéndola a cero, por ejemplo), a menudo es útil para distinguir entre lo que son datos del programa actual y lo que es basura del anterior.

Ejemplo:

```
-f 100 12f 'DATOS'
-d 100 12f
0B1B:0100  44 41 54 4F 53 44 41 54-4F 53 44 41 54 4F 53 44   DATOSDATOSDATOSD
0B1B:0110  41 54 4F 53 44 41 54 4F-53 44 41 54 4F 53 44 41   ATOSDATOSDATOSDA
0B1B:0120  54 4F 53 44 41 54 4F 53-44 41 54 4F 53 44 41 54   TOSDATOSDATOSDAT
-
```

D.2.2.9 Editar: E

```
Parámetro: dirección [valor o lista de valores separados
                       con espacio]
```

Se utiliza para ingresar datos o instrucciones (como código de máquina) directamente en ubicaciones de memoria específicas. Se puede usar de dos formas:

1. Proporcionando sólo la dirección: *DEBUG* muestra el valor del byte almacenado en la dirección proporcionada y nos da la oportunidad de cambiarlo escribiendo el valor en hexadecimal- A partir de este punto se ofrecen tres alternativas:
   a) No queremos cambiar el valor: pulsamos la barra espaciadora para pasar a la siguiente dirección.
   b) Queremos cambiar el valor: escribimos el valor y pulsamos la barra espaciadora para pasar a la siguiente dirección guardando los cambios.
   c) Conservamos el valor o lo cambiamos, pero ya terminamos nuestro trabajo: si lo queremos cambiar, escribimos el valor; si no, lo dejamos en blanco. Para terminar, pulsamos *Intro*. El valor se conserva o sobrescribe dependiendo el caso, pero se vuelve a la línea de comandos de *DEBUG*.
2. Proporcionando la dirección y el valor(es): *DEBUG* cambia el valor de la dirección indicada y vuelve de inmediato a la línea de comandos.

Puede verificar sus cambios usando el comando D (Volcar).

Ejemplos:

Cambiaremos un sólo byte en la ubicación *CS:FFCB* del valor que tenga a *D2*:

```
-e ffcb d2
```

En los siguientes dos ejemplos se muestra que, tanto comillas dobles (") como sencillas ('), pueden utilizarse para introducir valores

ASCII en una localidad de memoria dada. Al permitir ambas formas de delimitadores se puede incluir las comillas dentro de la cadena a usar de relleno:

```
-e 200 'A una "cadena ASCII-Z" siempre le sigue '
-e 22a "un byte cero ('00h')." 00
```

D.2.2.10 Ir a: G

```
Parámetro: [=dirección] [direcciones]
```

Este comando se utiliza para ejecutar un programa y establecer puntos de interrupción en el código en el mismo. Como vimos en el primer ejemplo del comando *EDITAR*, la opción 'dirección' se usa para indicarle a *DEBUG* una ubicación de inicio. Si usa 'g' por sí sola, la ejecución comenzará en la dirección a la que apunten los registros *CS:IP*. Se pueden establecer puntos de interrupción opcionales (lo que significa que el programa se detendrá antes de ejecutar el código en cualquiera de estas ubicaciones) de hasta diez direcciones, simplemente enumerándolas en la línea de comando.

Requisitos: Los puntos de interrupción sólo se pueden establecer en una dirección que contenga el primer byte de un código de operación 8088/8086 válido. Así que no se sorprenda si al elegir una dirección arbitraria el programa nunca se detiene; especialmente si está intentando depurar un programa que contiene códigos de operación que *DEBUG* no puede entender (instrucciones que 'requieren' una CPU por encima de un 8088/8086).

Precaución: *DEBUG* reemplaza las instrucciones originales de las direcciones de la lista de interrupciones con CCh (una interrupción 3; INT 3). Las instrucciones en estas ubicaciones se restauran con las instrucciones originales únicamente si los puntos de interrupción se encuentran. Si *DEBUG* no se detiene en ningún punto de interrupción, ¡todos sus puntos de interrupción todavía estarán habilitados! Por lo tanto, nunca guarde el código tal y como está almacenado en memoria, a menos que esté seguro de que *DEBUG* ha alcanzado todos los puntos de interrupción y reemplazado las instrucciones por sus originales.

Guardar una copia de seguridad antes de definir los puntos de interrupción suele ser una mejor idea.



D.2.2.11 Ensamblar: A

```
Parámetro: [dirección]
```

Crea un código ejecutable en la memoria de la máquina. Existen dos casos específicos para el uso del comando:

1. Si se omite el parámetro, el ensamblado se iniciará en la localización especificada por *CS:IP*, usualmente 0100H, que es la localización donde deben iniciar los programas con extensión .COM. Ésta será la localización que utilizaremos debido a que *DEBUG* sólo puede crear este tipo específico de programas.
2. Si el parámetro se omite, pero ya se utilizó anteriormente este comando, el ensamblaje comienza a partir de la última dirección utilizada.
3. Si se especifica la dirección, el ensamblaje comienza a partir de la dirección especificada.

DEBUG ensambla el conjunto de instrucciones del lenguaje ensamblador 8086/8088 (y 8087) que se ingresan comenzando en la dirección CS:0100 (o la dirección especificada). Aunque no se reconocen instrucciones ni etiquetas de macro, puede usar las pseudo instrucciones DB y DW. Puede, así, usar dichos códigos para ingresar datos ASCII de la siguiente forma:

```
DB 'Esto es una cadena', 0D, 0A
```

Es posible, también, comentar su programa usando el punto y coma (;). Toda cadena después de este carácter será ignorada.

El comando *A* recuerda la última ubicación donde se introdujeron datos, por lo que el uso de comandos *A* sucesivos (cuando no se especifica una dirección) siempre comenzarán en la siguiente dirección de las cadenas de instrucciones ensambladas. Este aspecto del comando es similar al comando *VOLCAR* que recuerda la ubicación de su último volcado de memoria (si no se especifica una nueva dirección).

El proceso de ensamblaje se detendrá después de ingresar una línea en blanco.

Ejemplo (escriba los caracteres en negrita):

```
-a
xxxx:0100 jmp 126
xxxx:0102 ;Las siguientes 2 instrucciones proveen los datos
xxxx:0102 db 0d,0a,'Un ejemplo de programa en ensamblador'
xxxx:0129 db 0d,0a,'$'
xxxx:012C xor ax,ax ; forma rápida de limpiar un registro
xxxx:012E mov ah,9
xxxx:0130 mov dx,102
xxxx:0133 int 21
xxxx:0135 mov ax,4c
xxxx:0138 int 21
xxxx:013A
-g =100
Un ejemplo de programa en ensamblador
Program terminated normally
-
```

D.2.2.12 Desensamblar: U

```
Parámetro: [rango]
```

Desensambla las instrucciones de la máquina en código 8086. Sin no se usa el [rango] opcional, utiliza un desplazamiento de 100 como punto de partida; desensambla unos 32 bytes y luego, si se vuelve a utilizar nuevamente comando, recuerda el siguiente byte como el punto de partida. No son exactamente 32 bytes porque puede ser necesario terminar con un número impar de bytes mayor que 32, dependiendo del último tipo de instrucción que *DEBUG* tenga que desensamblar.

NOTA: El usuario debe decidir si los bytes que *DEBUG* desensambla son instrucciones 8086, sólo datos o un conjunto de instrucciones x86 más recientes (como las de los *CPU* 80286, 80386 hasta la última generación que produce Intel; que pueden estar más allá de la capacidad de comprensión de DEBUG)

Ejemplos (en negrillas lo que debe introducir):

```
-u 126 133
0B1B:0126 64            DB      64
0B1B:0127 6F            DB      6F
0B1B:0128 720D          JB      0137
0B1B:012A 0A24          OR      AH,[SI]
0B1B:012C 31C0          XOR     AX,AX
0B1B:012E B409          MOV     AH,09
0B1B:0130 BA0201        MOV     DX,0102
0B1B:0133 CD21          INT     21
-
```

Si después de introducir el ejemplo de la instrucción *ENSAMBLAR* usa

```
-u 100,133
```

Verá el mismo programa que introdujo (los datos no son restituidos como códigos ASCII).

Si requiere primero cargar un programa, siga este procedimiento:

1. Muévase al directorio de Windows en donde reside el programa a desensamblar (instrucción chdir o cd). Esto facilita la tarea.
2. Ejecute *DEBUG*. Aparecerá el guion indicador de línea de comandos propio del desensamblador que indica que está listo a recibir órdenes.
3. Indicaremos, a continuación, el nombre del programa y lo cargaremos en memoria (ver el ejemplo de programa en el comando *N*):

```
-n dos_win.com
-l
```

4. El siguiente y último paso consiste en indicarle a *DEBUG* que lo desensamble:

```
- u
0BAE:0100 EB3B          JMP     013D
0BAE:0102 0D0A45        OR      AX,450A
0BAE:0105 7320          JNB     0127
0BAE:0107 63            DB      63
0BAE:0108 6F            DB      6F
0BAE:0109 7272          JB      017D
0BAE:010B 65            DB      65
0BAE:010C 63            DB      63
0BAE:010D 746F          JZ      017E
…
-
```

Notará que aparentemente sólo la 1era instrucción parece coincidir. Esto es porque a partir de la *cs:102* y hasta la *cs:013D* llenamos la memoria con datos. Si usamos una vez más la instrucción *U* llegaremos al punto en que las instrucciones coinciden:

```
-u 013d
0BAE:013D BA0201        MOV     DX,0102
0BAE:0140 B409          MOV     AH,09
0BAE:0142 CD21          INT     21
0BAE:0144 B8014C        MOV     AX,4C01
…

-
```

Ahora podemos ejecutar el programa, revisar los registros y salir:

```
-g 100
AX=0000 BX=0000 CX=0049 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=0BAE ES=0BAE SS=0BAE CS=0BAE IP=0100 NV UP EI PL NZ NA PO NC
0BAE:0100 EB3B      JMP   013D
-g
Es correcto ejecutar este programa desde DOS o Windows
Program terminated normally
-q
c:\ejemplos_debug>_
```

D.2.2.13 Entrada: I

```
Parámetro: puerto
```

El uso de comandos de E/S mientras se ejecuta Windows 9x/Me es simplemente poco confiable. Esto es especialmente cierto cuando se intenta acceder directamente a discos duros. En Windows NT/2000/XP, los comandos de E/S son sólo una emulación; así que no confíe en ellos. Aunque en el ejemplo siguiente todavía funciona en Win2000/XP, lo más probable es que utilice algún código WinAPI para mostrar lo que hay en el área del reloj de Windows; no directamente de un chip RTC[67].

Hace mucho tiempo (cuando *DOS* era el único sistema operativo para *PC*), había docenas de programas escritos en *BASIC* que usaban comandos de E/S para manejar tareas a través de puertos paralelos y seriales (por ejemplo, para cambiar la tipografía utilizada por una impresora o los valores en los registros de control de un módem). Bajo *DOS* real, todavía se pueden usar para comunicaciones directas con teclados o chips de control de una unidad de disco duro junto con muchos otros dispositivos de hardware.

A continuación, se muestra un ejemplo de cómo leer las horas y los minutos del "reloj en tiempo real" (RTC) de una computadora (no incluya ni la 🡐 ni el comentario que le sigue, solo lo que está en negrillas):

```
-o 70 04 🡐 Verifica la hora.
-i 71
```

[67] Real Time Clock (Reloj en Tiempo Real).

```
20       ← 20 horas (o las 8PM)
-o 70 02 ← Verifica los minutos.
-i 71
35       ← 35 minutes
```

D.2.2.14 Salida: O

```
Parámetro: Byte del puerto
```

Vea los comentarios y ejemplo en el comando ENTRADA.

D.2.2.15 Cargar Archivo: L

```
Parámetro: [dirección] [unidad] [sector] [cuenta]
```

Este comando cargará en la memoria el número seleccionado de sectores desde la unidad lógica de cualquier disco bajo el control de MS-DOS o Windows.

- La dirección es la ubicación en la memoria de donde se copiarán los datos (use sólo 4 dígitos hexadecimales para mantenerlos dentro de la memoria asignada a *DEBUG*).
- El número de unidad se asigna como: 0=A:, 1=B:, 2=C:, etc.
- El primer sector inicia su cuenta desde cero hasta el sector más grande en el volumen y finalmente,
- La cuenta específica, en hexadecimal, el número total de sectores que se copiarán en la memoria (por lo que un disquete con 0 a 2.879 sectores sería: 0 a B3F en hexadecimal).

Vea también los comandos *A*, *L*, *N*, y *U*.

D.2.2.16 Mover: M

```
Parámetro: rango de dirección separados por comas
```

Este comando realmente debería llamarse: COPIAR (y no Mover), ya que en realidad copia todos los bytes dentro del rango especificado a una nueva dirección.

Ejemplos:

```
m 7c00 7cff 600
```

Copia todos los bytes ente la dirección 7C00 y 7CFF (inclusive) a la dirección 0600 y las siguientes.

```
m 100 2ff 70
```

Este segundo ejemplo muestra que es muy fácil sobrescribir la mayor parte de la fuente, desde la que está copiando, usando este comando. Aparentemente, DEBUG almacena los bytes de origen en otro lugar antes de escribirlos; de otra forma, este ejemplo causaría un problema cuando comienza a sobrescribir lo que aún no ha copiado. Esto copia 512 bytes entre las direcciones 100h y 2FFh (inclusive) al desplazamiento 0070 sobrescribiendo los primeros 368 bytes en el proceso.

D.2.2.17 Nombrar: N

```
Parámetro: [[disco:][ruta]nombre_archivo [lista_de argumentos]]
```

Este comando se puede usar para cargar archivos en la memoria de *DEBUG* después de haber iniciado el programa, pero su función principal es crear un nuevo archivo bajo el control del sistema operativo en el que *DEBUG* puede *GUARDAR* (*W*) luego sus datos o programas.

Normalmente, cuando desea 'depurar' un archivo, debe iniciar *DEBUG* con un comando como éste: C:\WINDOWS>debug prueba.com. Pero también es posible cargar un archivo en la memoria de *DEBUG* usando el comando *N* seguido del comando *L* (sin parámetros) como en el siguiente ejemplo:

```
-n c:\temp\prueba.com
-l
```

lo que cargará el archivo prueba.com en la memoria de *DEBUG* comenzando en la ubicación CS: 0100 (no es posible especificar ninguna otra ubicación cuando se usa el comando *L*). En este punto puede usar el comando *U*.

El comando *N* hace que sea fácil guardar datos, o un programa ensamblador creado en *DEBUG*, en un archivo en su disco duro.

Por ejemplo, estos comandos (en negrita; junto con las respuestas de *DEBUG*):

```
-n c:\temp\DOS_WIN.com
-a 100
cs:0100 jmp 13D
cs:0102 db 0d,0a,"Es correcto ejecutar este "
cs:011E db "programa en DOS o Windows."
cs:013A db 0d,0a,24
cs:013D mov dx,102
cs:0140 mov ah,9
cs:0142 int 21
cs:0144 mov ax,4c01
cs:0147 int 21
```

```
cs:0149
-rcx
CX 0000
:49
-w
Writing 00049 bytes [ 73 bytes in decimal ]
-q
```

Lo que creará un archivo de 73 bytes llamado DOS_WIN.com en la carpeta C:\temp. Sin embargo, los nombres de los archivos están limitados a los ocho caracteres de DOS más tres para la extensión (esto se llama a menudo un nombre de archivo 8.3).

Para el caso contrario (desensamblar) siga el procedimiento delineado en el comando *U*.

Ejercicio

Siga los pasos anteriores para ensamblar y guardar este programa en DEBUG y luego use DEBUG para depurarlo. Utilice el comando *P* (roceder) para recorrer la mayoría de las instrucciones, ya que esto evitará que ingrese accidentalmente a una instrucción *INT* (errrupción). Si alguna vez usa el comando *T* (Rastreo) en un *INT*, terminará dentro de nidos de rutinas y subrutinas del *BIOS* que a menudo hacen que *DEBUG* falle.

D.2.2.18 Registros: R

```
Parámetro: [registro o banderas]
```

Este comando muestra el valor de uno o todos los registros del CPU además de la siguiente instrucción a ejecutar.

Hay tres casos posibles de su uso:

1. Uso sin parámetros: muestra el contenido de todos los registros de la CPU. DEBUG muestra, a continuación, la línea de comandos invitándole a introducir un nuevo comando.
2. Uso con un parámetro (*R* seguido del nombre del registro con o sin espacio) mostrará el contenido del registro especificado cambiando el indicador de comandos de “-“ a “:” invitándole a que cambie el valor de dicho registro. Deje en blanco o escriba el nuevo valor y pulse Intro para volver a la línea de comandos de *DEBUG*.
3. Uso con un parámetro (*R* seguido de *f* o *F* con o sin espacio) mostrará el contenido de las banderas indicador de comandos a “–” invitándole a que cambie el valor de cualquier bandera (una o varias separándolas con espacios).

Deje en blanco o escriba el nuevo valor(es) y pulse Intro para volver a la línea de comandos de *DEBUG*. Use la tabla *D.2* para auxiliarse.

Ejemplos:

1. Cambio de registros

```
-r
AX=0000 BX=0000 CX=1437 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=0ED8 ES=0ED8 SS=0ED8 CS=0ED8 IP=0100 NV UP EI PL NZ NA PO NC
0ED8:0100 E90E01 JMP 0211
-
```

2. Cambio de banderas

```
-rf
NV UP EI PL NZ NA PO NC  -zr cy
-rf
NV UP EI PL ZR NA PO CY  -
-
```

La siguiente tabla muestra la designación usada para los registros de banderas del *CPU:*

Tabla D.2

Banderas del CPU		
Nombre	Fijar	Limpiar
Saturación (sí/no)	OV	NV
Dirección (sumar/restar)	DN	UP
Interrupción (habilitar/deshabilitar)	EI	DI
Signo (negativo/positivo)	NG	PL
Cero (sí/no)	ZR	NZ
Acarreo auxiliar (sí/no)	AC	NA
Paridad (par/impar)	PE	PO
Acarreo (sí/no)	CY	NC

D.2.2.19 Rastrear: T

Parámetro: [=dirección] [número]

El comando *T* se usa para rastrear (recorrer) las instrucciones de la *CPU* una a la vez. Si ingresa este comando sólo sin parámetros, recorrerá solo una instrucción comenzando en la ubicación especificada por los registros *CS:IP*; se detendrá la ejecución del programa y luego mostrará todos los registros de la *CPU* además de la versión sin ensamblar de la siguiente instrucción a ser ejecutada; este es el modo "predeterminado" del comando *RASTREAR*. Sin embargo, digamos que desea que *DE-BUG* rastree y ejecute siete instrucciones que comienzan en la dirección CS:0205; para ello, se debería usar:

```
-t =205 7
```

Recuerde que el valor para el número de instrucciones a ejecutar debe darse en hexadecimal al igual que todos los demás valores usados en *DEBUG*. Dado que el comando T usa el "modo de seguimiento de hardware" de la *CPU*, es posible recorrer las instrucciones en una *ROM* o Memoria de sólo lectura.

D.2.2.20 Continuar: P

```
Parámetro: [=dirección] [número]
```

Continuar actúa exactamente igual que el comando T (Rastrear) para la mayoría de los tipos de instrucciones. La excepción es que *P* ejecutará inmediatamente todas las instrucciones (en lugar de pasar por cada una de ellas) dentro de cualquier subrutina *CALL*, *LOOP*, una instrucción tipo *REP*etir cadena o cualquier instrucción de *INT*errupción programática. Esto significa que no tiene que recorrer todas las instrucciones de una subrutina o llamada *INT* si usa este comando (*P*). En general el comando *P* será el usado con más frecuencia para depurar programas, y *T* (rastrear) solo se usará para ingresar a una subrutina o quizá verificar la lógica de las primeras iteraciones de una instrucción *LOOP* o *REP*.

El uso del comando *T* (*Rastreo*) puede ser incómodo si no se quiere depurar el código de las rutinas de interrupción o si ya se sabe el código que hay en las subrutinas y tan sólo le interesa seguir avanzando sin entrar en ellas. En estos casos se usa *P*.

D.2.2.21 Guardar: W

```
Parámetro: [dirección] [disco] [Primer_sector] [número]
```

No juegue con el comando W en *DEBUG* pues al tratar de crear nuevos archivos en un sector u otros usos no



controlados y experimentales puede destruir partes importantes de archivos valiosos o secciones importantes de su sistema operativo.

Al intentar escribir directamente a su disco duro usando un número de sector lo más probable es que pierda información (en el mejor de los casos) o corrompa la estructura de su disco duro (en el peor de los casos).

El comando Guardar (*W*) se usa a menudo para escribir un programa en su disco duro desde *DEBUG*. Pero la única forma segura de hacerlo, especialmente en Windows, es permitir que el sistema operativo decida dónde crear físicamente ese archivo en el disco. Esto se hace usando primero el comando Nombrar (*N*) para configurar una ruta opcional y un nombre de archivo para el nuevo fichero (o para sobrescribir uno ya existente). *DEBUG* comenzará a guardar automáticamente el programa o los bytes de datos desde la dirección 0100 del segmento de 64 KB que el sistema operativo le asignó. El único otro requisito es establecer el tamaño del archivo que desea escribir colocando el número total de bytes en los registros combinados *BX* y *CX* antes de ejecutar el comando *W*. Use el comando *R*egistros para cambiar el valor almacenado en *CX*.

D.3 Errores

Cuando *DEBUG* no sabe interpretar un comando, muestra un mensaje de error y un indicador "^" debajo de la posición del comando donde está el error. A continuación, un ejemplo:

```
-g 878788
       ^Error
```

D.4 Ejecutando un Archivo de Comandos

La mayoría de los comandos de *DEBUG* ejecutan una acción y vuelven al indicador de su propia línea de comandos a esperar una nueva orden. Si el resultado del comando es largo, como puede ser mostrar un trozo grande de código, puede detenerse pulsando *Ctrl→Pausa* o interrumpirse con *Ctrl→C* para volver a la línea de comandos.

Una característica poco conocida, es que *DEBUG* puede aceptar entradas desde un fichero previamente editado con comandos. Este archivo puede ser un simple fichero de texto *ASCII* en el que

cada comando esté separado del anterior por un *Intro*. Después del último, que debe ser una *Q* para salir de *DEBUG*, es conveniente dejar una línea en blanco pulsando *Intro* dos veces. Las líneas pueden contener comentarios. Cualquier cadena de caracteres a partir del punto y coma (;) hasta el final de la línea, será ignorado.

```
; esto es un comentario
D ; aquí se mostrará algo...
```

Suponiendo que tengamos un fichero de comandos llamado “ordenes.txt”; puede utilizarse como entrada para *DEBUG* mediante un comando de redirección de la siguiente forma:

```
DEBUG < Ordenes.txt
```

También puede conseguirse que el programa redireccione la salida hacia un fichero que puede ser inspeccionado más tarde. Aunque tiene la dificultad de tener que trabajar "a ciegas", puede ser de utilidad en determinadas circunstancias. Por ejemplo, cuando se desea un volcado de determinadas zonas de la memoria. En el caso anterior podría obtenerse un fichero llamado “resulta.txt” (no olvide la regla 8.3) con el siguiente comando:

```
DEBUG < Ordenes.txt > Resulta.txt
```

D.5 Ejercicios

D.1 Despliegue un texto en la pantalla. Comandos utilizados: *A*, *G*, *N*, *R*, *W*, *Q*.

Notas para este ejercicio:

- No olvide comenzar a ensamblar en la dirección 010016 (*A 100*)
- Verifique las direcciones de inicio y fin de ensamblado
- Verifique si *DX* y la dirección de la cadena de despliegue corresponden
- Apóyese en el ejemplo del comando *N* para ver cómo se guarda un archivo. Si modifica el texto a desplegar tenga cuidado con lo que escribe en *CX*.

5. Ensamble el siguiente código (*A*):

```
MOV AH,09              ; 09h en AH
MOV DX,0109            ; 0109h en DX
; DX tiene la dirección de la cadena a mostrar en pantalla
INT 21                 ; esta interrupción + el código 09h
                       ; de AX muestra una cadena en pantalla
```

```
INT 20                   ; Finaliza el programa
DB 'Hola a todos',D,24 ; Cadena a mostrar, la D simboliza el 13h
                         ; o retorno de línea y el 24h (ASCII de
                         ; $) termina la cadena a mostrar.
                         ; Esta instrucción debe empezar en 0109h
```

6. Ejecútelo (*G*)
7. Guárdelo: póngale un nombre (*N*; no olvide la regla 8.3), asigne el número de bytes a escribir (observe la última línea del código del programa y use *RCX*) y, finalmente, escríbalo de memoria al archivo previamente nombrado (*W*).
8. Salga de *DEBUG* (*Q*).

D.2 Rastree la ejecución del código del ejercicio *D.1* y verifique el contenido de los registros. Comandos utilizados: *T*.

D.3 Realice en secuencia las siguientes operaciones (verifique sus resultados con *R*):

- Coloque el valor FFFF en el registro *AX*
- Coloque el valor FFFF en el registro *BX*
- Sume los valores de AX y *BX* y coloque el resultado en *AX*
- Coloque el valor 0001 en el registro *AX*
- Disminuya el valor del registro *AX*
- Incremente el valor del registro *AX*
- Substraiga 2h del valor del registro *AX* y coloque el resultado en *AX*
- Coloque el valor 70h en el registro *AH*
- Coloque el valor 50h en el registro *BH*
- Sume el valor de *AH* al de *BH* y coloque el resultado en el registro *AH*

Respuestas:

```
MOV AX,FFFF
MOV BX,FFFF
ADD AX,BX
MOV AX,0001
DEC AX
INC AX
SUB AX,0002
MOV AH,70
MOV BH,50
ADD AH,BH
```

D.3 Abra una segunda ventana en la pantalla y muestre un carácter con su atributo. Comandos usados utilizados: *A*, *G*, *N*, *R*, *W*, *Q*.

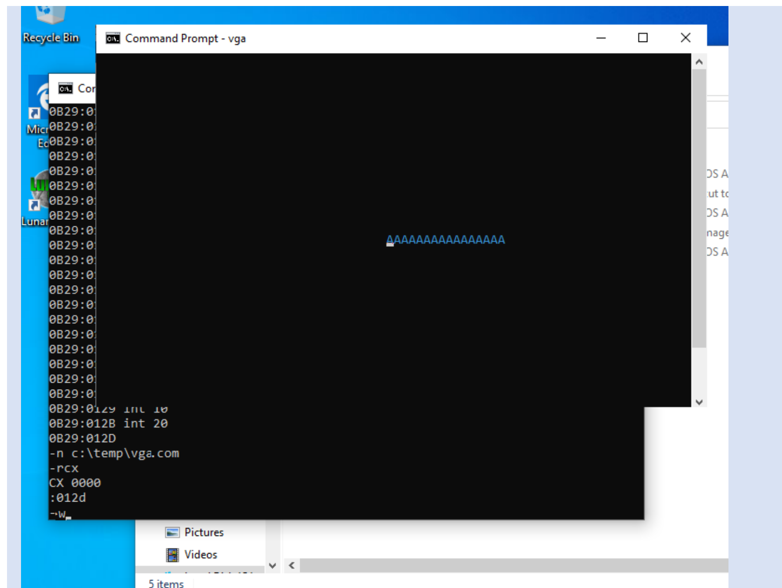
Ensamble el siguiente programa:

```
; Abre 2da pantalla
MOV AH,05 ; muestra pág.
MOV AL,01 ; Abre 2da pantalla
INT 10    ; int. de BIOS para mostrar pág. solicitada
;
; coloca cursor en medio
MOV AH,02 ; función
MOV BH,01 ; núm. pág. a mostrar
MOV DH,0C ; línea
MOV DL,27 ; columna
INT 10    ; posiciona cursor
;
; dibuja una A en cian 16 veces
MOV AH,09 ; dibuja carácter
MOV AL,41 ; una A ASCII
MOV BH,01 ; pág. 1
MOV BL,03 ; cian
MOV CX,10 ; 16 veces
INT 10    ; muestra el carácter con su atributo
;
; Espera respuesta del usuario para continuar
MOV AH,06 ; interfaz con usuario
MOV AL,00;
MOV DL,FF ; leer del teclado y guarda el AL
CMP AL,00 ; continua si no hay respuesta
JZ 011D   ; 011D es la dir. de la inst. del bucle
          ; que corresponde a MOV AH,06
;
; regresa la pantalla inicial
MOV AH,05 ; pág. original
MOV AL,00 ; núm. pantalla
INT 10    ; muestra pág. solicitada
INT 20    ; fin del programa; núm. de pág. a mostrar
;
```

Si desea usar otro código de color auxíliese de la siguiente tabla (*D.3*):

Tabla D.3					
Código de colores					
01h	Azul	02h	Verde	03h	Cian
04h	Rojo	05h	Magenta	06h	Café (marrón)
07h	Blanco	08h	Gris	09h	Azul claro
0ah	Verde claro	0bh	Cian claro	0ch	Rojo claro
0dh	Magenta claro	0eh	Amarillo	0fh	Blanco intense

Déle un nombre a su programa y guárdelo. Salga de *DEBUG* y ejecútelo. La imagen siguiente le muestra lo que debe obtener:



D.4 Introducir el siguiente código para la suma de 2 números:

```
Código de Máquina: 01D8
Asignar a AX=5
Asignar a BX=9
```

D.5 Introducir el siguiente código para la resta de 2 números:

```
Código de Máquina: 29F8
Asignar a AX=5
Asignar a BX=A
```

D.6 Introducir el siguiente código para la multiplicación de 2 números:

```
Código de Máquina: F7E3
Asignar a AX=5
Asignar a BX=D
```

D.7 Introducir el siguiente código para la división de 2 números:

```
Código de Máquina: F7F3
Asignar a AX=A
Asignar a BX=4
```

D.8 Utilizando los ejercicios del *D.4* al *D.7* realice un programa que genere secuencialmente la suma, resta, multiplicación y división, en ese orden. Los valores para estos ejercicios son tomados de los registros *AX* y *BX* y son los que usted escoja.

E Bibliografía

Alcalde San Miguel, Pablo. ***Electrónica Aplicada 2nda Edición***. Editorial Paraninfo (2016).

Bartee, C. Thomas. ***Digital Computer Fundaments***. McGraw Hill International Ed. (1985). Capítulo 7.

Byte Magazine. Octubre de 1992, 1990 a 1993 y varias más.

Desoer, Charles; A. Kuh; Ernest S. ***Basic Circuit Theory****.* Mc-Graw Hill International Ed. (2009).

Donovan, John J. ***Systems Programming*** McGraw Hill International Student Edition (1985).

Eggebrecht, Lewis. ***Interfacing to the IBM Personal Computer***. Segunda edición Howard W. Sams & Co. (1990).

Enciclopedia Británica 2015 Ultimate*.* Edición digital en DVD, (2016).

Fink, G. Donald. ***Electronics Engineers' Handbook***. McGraw Hill (1975).

Forest, M. Mims III. ***Engineer's Mini-Notebook 555 Timer IC Circuits***. Radio Shack fourth edition (1996).

Gonick, Larry. ***The Cartoon Guide to Computer Science***. Harper & Row Publishers NY (1983).

Hayes, John P. ***Digital System Design and Microprocessors***. McGraw Hill International (1987) Ed. 3era edición.

Hermoso, Donate Antonio. ***Electrónica Aplicada***. Ediciones Técnicas Marcombo (2013).

Heywood, Stephen A. ***The 8086 An Architecture for the Future***. Revista Byte (1983) junio, julio y agosto.

Hill, J. Frederick y Peterson, R. Gerald. ***Teoría de Conmutación y Diseño Lógico***. Limusa México (1984) cuarta reimpresión.

Horowitz, Paul y Hill, Winfred. ***The Art of Electronics Third Edition***. Cambridge University Press (2015).

Jung.Walt G. ***IC Timer CookBook***. Sams Publishing (983) 2nd Edition.

Karnaught, M. ***The Map Method for Synthesis of Combinational Logic Circuits*** AIEE (1953) 72 pg. 593-598 (1953).

Karnaught, M. ***The Map Method for Synthesis of Combinational Logic Circuits***. Journal of Symbolic Logic Nº 197 (junio 2014).

Lafore, Robert. ***Assembly Language Primer for the IBM PC & XT***. The Waite Group (1984).

Lancaster Don. ***TTL Cookbook***. Sams Books (1991) 22ava impresión.

Lancaster, Don. ***CMOS Cookbook***. Sams (1997) segunda edición.

Lesea Austin, Zaks Rodnay. ***Microprocessor Interfacing Techniques***. Sybex (1984).

Llano Díaz Emiliano. ***Análisis y Diseño de Compiladores***. UNAM (2002).

Llano Díaz Emiliano. ***Ensamblador***. UNAM (2002).

Llano Díaz Emiliano. ***Telecomunicaciones y Teleproceso***. UNAM (1989).

Macro 86. Columbia Data Products, Inc. (1982).

McCluskey, E.J. ***Minimization of Boolean Functions***. Bell System Technological Journal (Nov. 1956) 35, num. 5 pág. 1417-1444.

Microsoft DOS ver 5.0. Microsoft Press (1992).

Microsoft Macro Assembler 5.0. Microsoft Press (1990).

MPC Operations Guide. Columbia Data Products, Inc. (1982).

MsMOS Handbook. Motorola, Inc. Phoenix, Arizona (1975).

Muller, Scott. ***Upgrading and Repairing PCs 22nd Edition***. Que Editions (2015).

Raghavendra, Krishnamurthy. ***Engineering Metrology and Measurements***. Oxford University Press (2013).

Scherz, Paul; Monk, Simon. ***Practical Electronics for Inventors 3dr Edition***. Mc-Graw Hill International (2013).

Stern, Lotar. ***Electronics Made Easy***. Popular Mechanics Company, Chicago USA (1963) Capítulo 1.

Taub, Schilling. ***Digital Integrated Electronics***. Mc-Graw Hill International Ed. (1987).

Texas Instruments Incorporated. ***Designing with TTL Integrated Circuits***. McGraw Hill International Student Edition (1981) 9nth edition.

The IBM Personal Computer MACRO Assembler. IBM press (1981).

The Random House Encyclopedia. (1990) Edición revisada.

Varios. ***An Introduction to Microcomputers***. Adam Osborn and Associates Inc. (1975) Capítulos 3, 4 y 5.

Varios. ***Electronic Engineer Handbook***. Philips, Holanda (1997).

Varios. ***Electronics Made Easy***, Popular Mechanic Company Chicago USA (1963) Cap. 1

Varios. ***Pequeña Enciclopedia Columbia***, Editorial Sudamericana Buenos Aires (1968).

Waite Mitchell, Pardee Michael. ***Microcomputer Primer***. Howard Sams & Co. (1980) Capitulos 3 y 4.

F Índice Alfabético

C

J

K

L

M

Q

R

S

T

U

V

W

X

Z